MASTER EN SCIENCES INFORMATIQUES

Vers un classifieur hybride

Évaluation de différentes stratégies d'intégration de ressources linguistiques au sein d'un modèle CRF

Eryilmaz, Serkan

Award date:
2013

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur

Faculté d'Informatique

Année académique 2012–2013

Vers un classifieur hybride. Évaluation de différentes stratégies d'intégration de ressources linguistiques au sein d'un modèle CRF.

Serkan Eryilmaz

Promoteur : _____

(Signature pour approbation du dépôt - REE art. 40)

Pr Dr Ir Pierre-Yves Schobbens

Mémoire présenté en vue de l'obtention du grade de

Master en Sciences Informatiques.

Je tiens tout d'abord à remercier mon promoteur, le professeur Pierre-Yves Schobbens, pour m'avoir permis de réaliser ce mémoire sous sa direction, mais aussi pour son encadrement, son suivi tout au long de l'année ainsi que ses conseils avisés,

Je remercie le docteur Patrick Watrin sans qui ce mémoire n'aurait probablement jamais été envisagé, pour son aide, ses conseils et sa patience infinie,

Je souhaite également remercier l'ensemble de mes collègues de la société EarlyTracks, pour leur soutien continu durant ces trois dernières années, et en particulier Stéphanie Weiser et Louis de Viron pour leur lecture attentive et leurs commentaires pertinents,

Je remercie Brice de Behault, chef d'une petite entreprise ambitieuse, pour son soutien et sa patience durant ces trois années où je l'ai malmené à chaque session d'examens,

Je tiens également à remercier mes camarades de classe et amis, Sélim Moulaï, Saffet Sengezer, Nicolas Verbeiren et Ludovic Verhoustraeten, ces années n'auraient pas pu être aussi agréables sans leur présence et leur bonne humeur,

Je souhaite remercier les membres du jury, madame le professeur Marie-Ange Remiche et monsieur le professeur adjoint Anthony Cleve, qui me font l'honneur de juger mon modeste travail,

Enfin, je remercie Marion Denisty, pour sa présence, sa patience, ses encouragements et sa lecture minutieuse de ce mémoire,

Merci à vous.

Résumé

Dans le domaine du traitement automatique du langage, la reconnaissance d'entités nommées (noms de personnes, de lieux, d'organisations, etc.) est un défi intéressant. Comme c'est le cas pour l'ensemble des applications liées au traitement automatique des langues (TAL), il existe deux approches concurrentes : l'approche linguistique et l'approche statistique. L'une et l'autre présentent des avantages et des inconvénients. Ces dernières années, un grand nombre de recherches menées dans ce domaine ont visé à fusionner les bénéfices des ressources linguistiques et des modèles statistiques au sein d'une approche hybride. Ainsi, plusieurs études ont envisagé l'intégration d'informations linguistiques au sein d'un classifieur CRF, particulièrement adapté à l'extraction des entités nommées. Dans ce mémoire, nous évaluons différentes stratégies d'intégration de ressources linguistiques (dictionnaires à large couverture, grammaires lexico-syntaxiques, ...). Cette évaluation systématique nous permet de définir une méthodologie optimale, et d'ainsi proposer un extracteur compétitif.

Abstract

In natural language processing (NLP), an interesting task is the recognition and classification of named entities (names of people, places, organizations, etc.). As is the case for most natural language processing tasks, there exists two different approaches: the linguistic approach and the statistical approach. Either one has its advantages and disadvantages. Over the last years, many research have been focused in bridging the gap between those two approaches by taking advantage of linguistic knowledge and statistical models in a hybrid approach. Some of those studies consisted in integrating linguistic knowledge in a CRF model, which are particularly good at named entity extraction and classification (NERC). In this master's thesis, we will evaluate different integration strategies (wide coverage dictionaries, lexico-syntactic grammars, ...). This systematic evaluation will allow us to define an optimal methodology in offering a competitive classifier.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'informatique, et en particulier Internet, a fondamentalement changé nos habitudes en terme de moyens de communication. Les courriels remplacent les courriers postaux, la messagerie instantanée remplace en partie la téléphonie, les blogs, les réseaux sociaux mais aussi et surtout les encyclopédies communautaires changent la manière dont nous recevons et transmettons les informations. La facilité et la gratuité de ces échanges ont mené à une réelle explosion de la quantité d'informations accessible à tous. Face à cette masse d'informations, les moteurs de recherche nous permettent de filtrer et éventuellement de trouver l'information la plus pertinente.

Les premières approches choisies par les moteurs de recherche sont textuelles, c'est-à-dire qu'elles identifient les documents contenant exactement les termes recherchés.

La recherche textuelle a cependant des limites que l'on rencontre lors de l'utilisation courante des moteurs de recherche. Pour illustrer, prenons par exemple la phrase de recherche « acheter des actions Apple » : quels documents le moteur de recherche doit-il retourner ? L'intention semble claire : on cherche à acheter

des actions boursières de la société Apple Inc. Mais les méthodes de recherche les plus naïves ne fonctionnent pas toujours : elles se basent seulement sur les fréquences d'apparition des mots dans les documents. Si un document contient la phrase « Acheter un Apple iPhone et le jeu Actions pour seulement 799 euros. », il se retrouvera dans les résultats, malgré qu'il ne corresponde pas aux attentes de l'internaute.

Les moteurs de recherche plus récents vont plus loin que de la simple recherche textuelle : ils permettent de définir un certain nombre de contraintes sur les termes ou sur le type de résultats attendu : la recherche est sémantiquement restreinte. Ces moteurs disposent, de manière générale, d'informations structurelles ou sémantiques sur les sources d'informations. Ces meta-informations peuvent être obtenues de plusieurs manières, mais deux moyens sont majoritairement utilisés : l'ajout manuelle de ces meta-informations (par l'auteur, par un employé, etc.) et l'extraction automatique de ces informations.

L'une des informations intéressantes dans un texte est la présence d'entités nommées : Ils représentent des noms de personnes, d'organisations, de lieux, de médicaments, etc.

Dans ce mémoire, nous nous intéressons à la tâche d'extraction automatique d'informations au sein de documents textuels, et en particulier à l'extraction automatique d'entités nommées (Named Entity Recognition and Classification, NERC) : dans l'exemple précédent, il s'agit de repérer « Apple » comme entité, mais aussi de lui attribuer l'étiquette « Entreprise ».

Une fois un tel système mis en place, il peut être utilisé afin d'enrichir les documents indexés dans le moteur de recherche. Cet enrichissement permet de filtrer les documents sur des critères sémantiques plus poussés. On peut également utiliser

ces informations dans le but d'effectuer du data mining, des analyses fréquentielles, etc.

Premiers travaux en extraction d'information

Bien que des travaux antérieurs aient été effectués, comme par exemple « Extracting company names from text » de Rau (1991), le domaine a réellement évolué de manière rapide et constante à partir de 1996, lors du MUC-6 (Grishman et Sundheim, 1996).

Depuis, le nombre de publications ne cesse d'augmenter et beaucoup d'évènements scientifiques sont organisés, comme le « HUB-4 (Chinchor *et al.*, 1998), MUC-7 et MET-2 (Chinchor, 1998), IREX (Sekine et Isahara, 2000), CONLL (Sang et Erik, 2002; Tjong Kim Sang et De Meulder, 2003), ACE (Doddington *et al.*, 2004) et HAREM (Santos *et al.*, 2006) » (Nadeau et Sekine, 2007).

Définition de la problématique

Dans les solutions existantes pour l'extraction des entités nommées, nous pouvons distinguer deux approches : les systèmes basés sur des règles (approche symbolique) et les systèmes basés sur l'apprentissage automatique (approche statistique). Les deux approches, fondamentalement différentes, ont chacune leur lot de difficultés et d'avantages.

En particulier, les techniques statistiques utilisées couramment possèdent un certain désavantage : un corpus d'entraînement lexicalement exhaustif doit être à disposition. En effet, cette approche nécessite un haut taux de couverture lexicale,

but qui n'est généralement réalisé qu'avec un grand corpus. Les approches symboliques quant à elles reposent sur un ensemble de règles construites manuellement, processus coûteux en ressources humaines tant l'effort nécessaire est grand pour atteindre une couverture syntaxique suffisante.

Pour contrer ces problèmes, nous montrons qu'une approche hybride, liant les avantages de ces deux approches, permet d'obtenir de meilleurs résultats en utilisant des dictionnaires à large couverture et des règles grammaticales pour augmenter la couverture lexicale d'une part, et d'autre part un corpus annoté, de taille raisonnable, pour assurer la reconnaissance de structures syntaxiques.

Structure du mémoire

Ce mémoire est subdivisé en deux parties principales : la première présente un parcours rapide de l'état de l'art et recense la littérature scientifique du sujet. La complexité intellectuelle se trouvant principalement dans l'approche statistique, cette partie se verra plus détaillée. La seconde partie présente la mise en place d'un système permettant d'extraire de l'information en utilisant une combinaison de techniques issues de l'approche symbolique et des approches statistiques. Nous montrons progressivement l'impact de toutes les ressources améliorant les performances d'un tel système. Enfin, nous abordons brièvement différentes perspectives d'amélioration du système.

Première partie

État de l'art

Introduction

Dans cette première partie, nous décrivons plus en détail les différentes formes d'analyses que nous effectuons sur le texte, afin de fixer le cadre du système à développer. Nous décrivons ensuite les deux approches traditionnelles du domaine : l'approche symbolique et les approches statistiques. Nous détaillons ensuite plus formellement le modèle probabiliste que nous utilisons pour la mise en place d'un système automatisé d'extraction d'informations : les champs aléatoires conditionnels (Conditional Random Fields ou CRF). Pour terminer, nous décrivons les différentes méthodes d'évaluation d'un tel système.

Chapitre 1

Etiquetage morphosyntaxique et extraction d'entités nommées

L'extraction d'entités nommées est une tâche pouvant s'appuyer sur des traitements linguistiques, tels que l'étiquetage morphosyntaxique. Cela consiste à annoter les mots d'une phrase en les analysant de deux manières possible : linéairement ou hiérarchiquement. Nous présentons dans ce chapitre les types d'annotation existants dans le cadre de l'étiquetage morphosyntaxique et l'extraction d'entités nommées. Les analyses hiérarchiques sont illustrées par des arbres.

1.1 Définitions

1.1.1 Morphosyntaxe

L'étiquetage morphosyntaxique consiste à effectuer l'analyse morphologique, l'analyse fonctionnelle, l'analyse syntaxique d'un texte ou une quelconque combi-

Pourtant$_{\text{ADV}}$, ces$_{\text{DET}}$ jets$_{\text{NC}}$ d'$_{\text{P}}$ éponges$_{\text{NC}}$ sont$_{\text{V}}$ doublement$_{\text{ADV}}$ paradoxaux$_{\text{ADJ}}$.

FIGURE 1.1 – Natures des mots

TABLE 1.1 – Lemmes

Flexion	Nature	Lemme
Pourtant	ADV	pourtant
ces	DET	ce
jets	NC	jet
d'	P	de
éponges	NC	éponge
sont	V	être
doublement	ADV	doublement
paradoxaux	ADJ	paradoxal

naison de ces analyses. Voici les définitions, non exhaustives, de chacune de ces analyses.

Analyse lexicale

Le but de l'analyse lexicale est de déterminer, pour chaque mot d'un texte, sa nature, c'est-à-dire la catégorie lexicale du mot mais aussi son *lemme*, c'est-à-dire sa forme canonique, telle qu'on la retrouve dans un dictionnaire. La figure 1.1 illustre un étiquetage des natures des mots.

Une fois la nature d'un mot obtenue, nous pourrons généralement déduire son lemme par l'utilisation de dictionnaires. Les différentes formes *fléchies* d'un mot sont appelées les *flexions*, et l'ensemble des flexions d'un mot représente donc toutes les formes que ce mot peut prendre. Le tableau 1.1 complète l'information de la figure 1.1 en ajoutant le lemme de chaque mot.

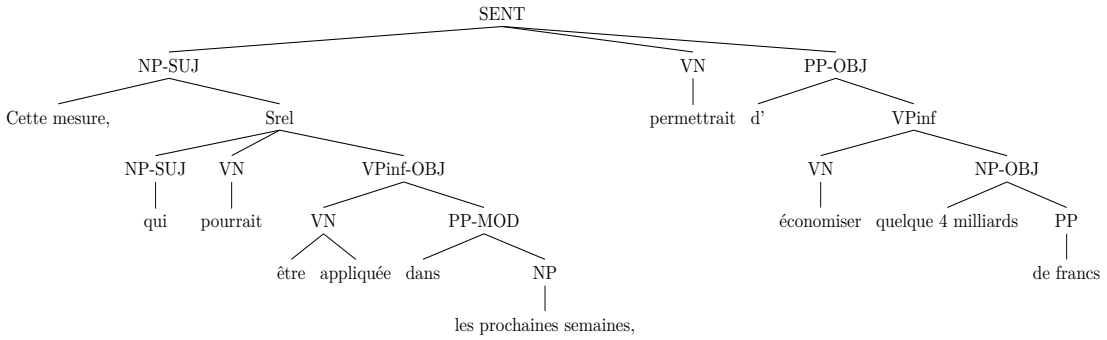

FIGURE 1.2 – Arbre syntaxique de la phrase.

Pourtant, ces jets d'éponges$_{\text{SUJ}}$ sont doublement paradoxaux$_{\text{COMPL-SUJ}}$.

FIGURE 1.3 – Fonctions de la proposition

Analyse fonctionnelle

Au delà de leur nature, les mots ou les ensembles de mots ont également une fonction dans la proposition. Les mots d'une proposition s'articulent autour d'un noyau, généralement le groupe verbal, et prennent une fonction dans la phrase (sujet, objet, complément, etc.). L'étiquetage fonctionnel de la proposition 1.1 est illustré par la figure 1.3.

Analyse syntaxique

Enfin, l'analyse syntaxique consiste à déterminer les différentes propositions qui forment la phrase. Une phrase comporte une ou plusieurs *propositions*, chacune composée de *constituants*. Les constituants sont des groupes de mots qui réalisent ensemble une fonction grammaticale. La figure 1.2 représente l'arbre syntaxique de la phrase. Cet arbre exprime les dépendances entre les constituants sous la forme d'une structure hiérarchique. Ces constituants se trouvent dans les feuilles de l'arbre.

Wolff $_{\text{PER}}$, actuellement un journaliste en Argentine $_{\text{COUNTRY}}$, jouait pour Del Bosque $_{\text{PER}}$ à la fin des années septantes au Real Madrid $_{\text{ORG}}$.

FIGURE 1.4 – Example de NERC (Sang et Erik, 2002)

1.1.2 Entités nommées

Une entité nommée est une unité (mot ou groupe de mots) qui réfère à une entité unique. Les noms de personnes, d'organisations, de lieux, d'espèces animales, de médicaments, mais aussi les montants, les unités de mesure, les dates et autres informations numériques sont des entités nommées. Leur identification et leur typage sont intéressants pour certaines applications, comme les moteurs de recherche ou les outils de data mining. Le terme est apparu lors d'une conférence sur le traitement automatique du langage, le MUC-6 (Grishman et Sundheim, 1996). La figure 1.4 montre un exemple d'étiquetage d'entités nommées.

Chapitre 2

Méthodes d'étiquetage morphosyntaxique et d'extraction d'entités nommées

Nous distinguons deux catégories d'approche pour l'étiquetage morphosyntaxique et l'extraction d'entités nommées : la première se base sur la construction manuelle d'un ensemble de règles et sont appelées les méthodes « symboliques » (ou « linguistiques »). La deuxième catégorie regroupe les méthodes d'apprentissage automatique, provenant du domaine de l'intelligence artificielle. Dans ce chapitre, nous décrivons rapidement en quoi consiste chacune de ces approches, ainsi que leur particularités.

2.1 Systèmes basés sur des règles

Les premières approches (Brill, 1993; Greene et Rubin, 1971) pour l'extraction d'informations sont axées sur la mise en place manuelle de règles syntaxiques, par des experts linguistes, avec l'aide d'outils d'analyse linguistique. Ces règles, décli-

nées sous formes de dictionnaires et de grammaires lexico-syntaxique [1] forment des ressources utilisables par un générateur d'automates. Les automates ainsi générés peuvent ensuite être appliqués à des textes non structurés afin de les annoter. Ces automates générant une sortie sont appelés *transducteurs*.

Cette approche permet de modéliser un domaine de manière très précise, et en fonction de l'effort, elle peut donner d'excellents résultats, surtout en terme de précision [2]. Cependant, l'effort requis est proportionnel à la taille du vocabulaire et les différentes formes d'expression du domaine visé.

Bien que le coût de développement initial est moindre que les approches statistiques, la difficulté réside dans la complétion des règles : décrire l'ensemble des formulations possible pour un langage naturel n'est généralement pas possible, et c'est pourquoi elles sont couramment utilisées pour une application dans un domaine réduit [3]. Enfin, si elles génèrent peu d'erreurs, ces méthodes sont cependant peu flexibles : les grammaires construites pour un certain domaine ne sont généralement pas pertinentes pour un autre domaine ni pour une autre langue.

2.2 Apprentissage automatique

L'apprentissage automatique (Machine Learning) consiste en une série de méthodes qui permettent à un système de converger vers une solution sans avoir recours à des méthodes manuelles, telles que l'élaboration de règles. Grâce à des techniques statistiques et probabilistes, cela consiste, dans notre cas, à modéliser

1. Par exemple, une grammaire peut contenir une règle lexico-syntaxique comme « ministre de **DET NC** » : cette règle permet de reconnaître des formulations telles que « minitre de l'éducation », « ministre de la justice », etc.
2. Précision au sens statistique : les méthodes d'évaluation sont discutées dans le chapitre 5.
3. Par exemple, il est envisageable d'utiliser exclusivement des règles si on se limite à reconnaitre les éléments d'une adresse dans un certain pays.

la structure du langage via une analyse de corpus. Ces techniques se divisent en plusieurs classes, soient les méthodes *supervisées*, les méthodes *non supervisées* et enfin les méthodes *semi-supervisées* :

Généralement, l'apprentissage automatique nécessite de posséder un corpus, soit un ensemble de textes cohérent pour un domaine à modéliser, à partir duquel on essaie d'obtenir des observations de manière automatique. En fonction de la méthode d'apprentissage, ce corpus doit être plus ou moins conséquent. Plus la taille requise du corpus est importante, plus le coût de développement est élevé.

2.2.1 Méthodes supervisées

L'apprentissage supervisé consiste en la création d'un modèle statistique à partir d'un corpus annoté. Un corpus annoté est un ensemble de textes représentatif du domaine concerné dans lequel la tâche que l'on veut automatiser a été faite manuellement. La figure 1.4 (page 20) montre un exemple concret de texte annoté pour la tâche d'étiquetage des entités nommées.

La qualité des résultats de l'automatisation dépend directement de la qualité du corpus. Il est très important de ne pas introduire d'erreurs dans le modèle d'apprentissage : un corpus annoté correctement, qui obtient par exemple une F-mesure [4] supérieure à 99%, aide grandement à l'obtention de bons résultats lors de l'application du modèle résultant de l'apprentissage.

4. La précision, le rappel et la F-mesure sont des mesures d'évaluation. Celles-ci sont discutées au chapitre 5, page 55.

2.2.2 Méthodes non supervisées

Les méthodes non supervisées sont un ensemble de techniques qui n'ont pas recours à des corpus annotés : un simple corpus suffit. Pour l'extraction d'information, ces approches servent à trouver des relations entre les mots à partir d'une certaine quantité d'observations. Une relation entre deux mots peut être, par exemple, « se terminent (tous les deux) en ER », ou encore « commencent par une majuscule », bien que ces relations peuvent être beaucoup plus complexes et subtiles. L'approche permet de partitionner les mots dans des ensembles homogènes. Ces groupements forment ce que l'on appelle des *clusters*.

Le but dans le NERC est d'étiqueter les mots, et ceci se traduira généralement par la création (automatique) de clusters par rapport à leur rôle sémantique (ex : un cluster de personnes, un cluster de sociétés, etc.). Cela amène des difficultés : le choix des différents paramètres que l'on va utiliser pour former ces clusters, comme par exemple leur taille et leur tolérance par rapport à un nouveau vocabulaire sont autant de problèmes complexes à résoudre.

Nadeau et Sekine (2007) listent quelques travaux qui ont été effectués dans la reconnaissance d'entités nommées avec des méthodes non supervisées : Alfonseca et Manandhar (2002) assignent à chaque *synset*[5] de WordNet[6] l'ensemble des mots avec lesquels ils apparaissent fréquemment et forment ainsi un *concept*. Pour la phase d'étiquetage, chaque mot est comparé à chaque concept afin de retrouver le couple le plus similaire et d'ainsi étiqueter ce mot comme faisant partie de ce concept. Shinyama et Sekine (2004) partent de l'hypothèse qu'une entité nommée apparaît de manière synchronisée dans des journaux et qu'on peut donc les détecter grâce à ce prédicat. Il reste cependant le problème du typage.

5. Synset : ensemble de termes qui partagent le même sens
6. http ://wordnet.princeton.edu/

2.2.3 Méthodes semi supervisées

Dans le cas où l'on possède un corpus annoté de trop faible taille pour obtenir de bons résultats via des méthodes purement supervisées, il est pertinent de pouvoir profiter des apports des méthodes supervisées dans le cadre d'utilisation de méthodes non supervisées. Cela consiste à combiner les méthodes supervisées et non supervisées dans le but d'obtenir de bonnes performances à partir d'un corpus partiellement annoté, ce qui allège la tâche humaine d'annotation. Ce processus, appelé *bootstrapping*, consiste à trouver des relations contextuelles au sein des éléments étiquetés afin d'annoter une plus grande partie du corpus en généralisant les prédicats contextuels et ensuite d'entraîner, à partir de ce corpus annoté en partie manuellement et en partie automatiquement, un modèle via une méthode supervisée.

2.3 Apprentissage automatique dans le cadre de l'extraction d'informations

Les techniques d'apprentissage automatique supervisé constituent aujourd'hui l'approche privilégiée pour l'étiquetage d'entités nommées (Nadeau et Sekine, 2007). Ces techniques permettent en effet d'entraîner des modèles permettant la reconnaissance de structures syntaxiques en généralisant des séquences d'observations à partir d'un corpus annoté.

Ces méthodes comprennent les modèles Markoviens, discutés dans le chapitre 3, mais également les arbres de décisions, les machines à vecteurs de support, les réseaux de neurones artificiels, etc.

Dans la suite de ce mémoire, nous voyons que ces méthodes sont quand même sujettes au problème de couverture lexicale. Nous tentons de minimiser ce problème via l'intégration de ressources linguistiques (tels que des dictionnaires à large couverture et des grammaires syntactico-sémantique).

Chapitre 3

Modèles probabilistes pour l'étiquetage de séquences

Les approches dominantes pour l'extraction d'entités nommées sont actuellement les méthodes supervisées, et en particulier les modèles probabilistes dont les modèles Markoviens font partie. Dans ce chapitre, nous parcourons les modèles de Markov cachés (HMM), puis les modèles de Markov à entropie maximale (MEMM), pour enfin conclure sur les champs aléatoires conditionnels (CRF). Ces trois modèles sont fortement liés car chacun entraîne le suivant : les CRF sont un dérivé des modèles de Markov à entropie maximale qui sont eux-mêmes dérivés des modèles de Markov cachés.

3.1 Modèle de Markov caché

3.1.1 Chaîne de Markov

Selon Rabiner (1989), une chaîne de Markov est un système qui peut être décrit, à tout moment, comme étant dans un certain état parmi un ensemble fini de N états, soient les états $S_1, S_2, \ldots, S_N$. Le système change régulièrement d'état en respectant un ensemble de probabilités de transitions. L'état du système à un instant t est dénoté q_t. Si l'on souhaite faire une description probabiliste complète d'un tel système, il faut spécifier l'état courant ainsi que l'ensemble des états précédents, ce qui est difficile et généralement infaisable. Cependant, dans le cas des chaînes de Markov de premier ordre, cette description est limitée à l'état courant q_t ainsi qu'à l'état précédent q_{t-1}. Cette limite est la force des modèles Markoviens et nous permet d'automatiser la modélisation probabiliste du processus caché.

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots] \tag{3.1}$$

$$= P[q_t = S_j | q_{t-1} = S_i] \tag{3.2}$$

L'ensemble des probabilités de transitions a_{ij} :

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], \qquad 1 \leq i, j \leq N \tag{3.3}$$

$$a_{ij} \geq 0$$

$$\sum_{j=1}^{N} a_{ij} = 1$$

Exemple

Supposons un système à 3 états, où chacun de ces états correspondent au temps qu'il fera durant la journée.

S_1 : Pluvieux

S_2 : Nuageux

S_3 : Ensoleillé

Avec la matrice A de probabilités de transitions d'états suivante :

$$A = \{a_{ij}\} = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

La probabilité qu'il fasse, pour les 7 prochains jours « soleil-soleil-pluie-pluie-soleil-nuages-soleil » sachant qu'il fait beau aujourd'hui est la résolution de la probabilité de la séquence d'observation $O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$ et peut se calculer comme suit :

$$
\begin{aligned}
P(O|Model) &= P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|Model] \\
&= P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \cdot P[S_1|S_1] \cdot P[S_3|S_1] \cdot P[S_2|S_3] \cdot P[S_3|S_2] \\
&= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\
&= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\
&= 1.536 \times 10^{-4}
\end{aligned}
$$

Où π_i dénote la probabilité de l'état initial, tel que :

$$\pi_i = P[q_1|S_i], \qquad 1 \leq i \leq N \tag{3.4}$$

Nous pouvons également calculer la probabilité que le système reste dans un certain état S_i pendant une certaine durée d. Ceci revient à calculer la probabilité de la séquence d'observation :

$$O = \{\underset{1}{S_i}, \underset{3}{S_i}, \underset{3}{S_i}, \ldots, \underset{d}{S_i}, \underset{d+1}{S_j} \neq S_i\}$$

Cette probabilité, appelée la densité de probabilité, peut être obtenue via la formule suivante :

$$P(O|Model, q_1 = S_i) = (a_{ii})^{d-1}(1 - a_{ii}) = p_i(d) \tag{3.5}$$

$$\begin{aligned} \bar{d}_i &= \sum_{d=1}^{\infty} dp_i(d) \\ &= \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) \\ &= \frac{1}{1 - a_{ii}} \end{aligned} \tag{3.6}$$

Selon ce modèle, le nombre de jours consécutifs de soleil attendu est donc de $\frac{1}{0.2} = 5$. Les nuages devraient perdurer pendant $\frac{1}{0.4} = 2.5$ jours et enfin, pour la pluie, $\frac{1}{0.6} = 1.67$ jours.

3.1.2 Définition des modèles de Markov cachés

Les processus dans le cadre d'une chaîne de Markov sont observables physiquement. En d'autres termes, il est possible d'observer l'état dans lequel se trouve le modèle. De plus, dans cet exemple, chaque état est lié à une seule observation possible. Un modèle caché de Markov est similaire, sauf que celui-ci n'est pas observable directement. La séquence d'observations est produite par un processus caché. Rabiner (1989) reprend l'illustration de Jack Ferguson (1960) comme suit :

Des urnes et des balles

Supposons qu'il y ait N urnes dans une chambre fermée, et que chaque urne contient des balles de couleur quelconque, parmi M couleurs possibles. Un génie est dans cette chambre, choisit une balle dans une urne et montre la balle en guise

d'observation, sans qu'il soit possible de voir le processus de sélection. Ensuite, il choisit une urne (la même ou une autre) en suivant un processus probabiliste qui ne prend en considération que l'urne de laquelle provient la balle précédemment sélectionnée. C'est ce processus caché que nous tentons de modéliser de manière probabiliste : le problème est donc de résoudre les différents paramètres du modèle.

1. Le nombre d'urnes, soit le nombre d'états N, avec chaque état $S = \{S_1, \ldots, S_N\}$, et l'état du système à l'instant t, q_t.
2. Le nombre de couleurs de balles différentes, soit le nombre de symboles d'observation possible M. Chaque symbole étant dénoté $V = \{V_1, \ldots, V_M\}$.
3. La probabilité de changer d'urne, soit la matrice A de probabilités de transitions d'états.

$$A = \{a_{ij}\} = P[q_{t+1} = S_j | q_t = S_i], \qquad 1 \leq i, j \leq N \tag{3.7}$$

4. La probabilité d'obtenir une certaine couleur dans une certaine urne, soit la matrice B de distribution de probabilités pour chacune des observations possibles dans l'état $S_j, B = \{b_i(k)\}$

$$\begin{aligned} b_i(k) &= P[V_k \text{ en } t, q_t = S_j], \\ & 1 \leq j \leq N \\ & 1 \leq k \leq M. \end{aligned} \tag{3.8}$$

5. La probabilité que chaque urne soit l'urne initiale, soit $\pi = \{\pi_i\}$ tel que

$$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N \tag{3.9}$$

On obtient ainsi le modèle complet, λ, noté comme suit :

$$\lambda = (A, B, \pi) \tag{3.10}$$

Le principal défi lors de la reconstruction d'un modèle de Markov est donc de trouver les paramètres de ce dernier. On va donc tenter de les approcher par un processus d'entraînement. Il n'existe pas d'algorithme permettant de résoudre les paramètres précisément, cependant, l'algorithme de Baum-Welch [7] (Baum *et al.*, 1970), tente de converger vers un modèle qui correspond le mieux aux observations réelles, c'est-à-dire en trouvant $\lambda = (A, B, \pi)$ maximisant $P = [O|\lambda]$.

3.1.3 Applications possibles

Une fois que les paramètres du modèles sont fixés, son application est triviale et permet de :

1. Générer des séquences d'observations possibles.
2. Déterminer si une séquence d'observations respecte le modèle du processus (et que ce même processus aurait donc pu générer).
3. Déterminer, pour une séquence d'observations donnée, la séquence d'états la plus probable qui aurait pu générer cette séquence d'observations.

Ce troisième point est le plus intéressant dans notre cas. En effet, si on associe les observations à des mots et les états à des classes d'entités (par exemple : *« Personne », « Ville », « Société », « Aucun »*), on peut étiqueter des séquences de mots (notamment des phrases). L'algorithme de Viterbi, décrit ci-après, nous permet précisément de le faire.

7. Également appelé forward-backward algorithm.

3.2 Algorithme de Viterbi

L'algorithme de Viterbi (1967) permet de trouver la séquence d'états la plus probable conformément à la séquence d'observations. Ci-dessous l'algorithme 1, illustré par la figure 3.1.

Entrées :
- L'ensemble des observations possibles $O = \{o_1, \cdots, o_N\}$,
- L'ensemble des états $S = \{s_1, \cdots, s_K\}$,
- Une séquence d'observations $Y = \{y_1, \cdots, y_T\}$,
- La matrice de transitions A de taille $K \times K$ tel que A_{ij} représente la probabilité de transition de l'état s_i à l'état s_j,
- La matrice de probabilité d'observations B de taille $K \times K$ tel que B_{ij} représente la probabilité de l'observation o_j à l'état s_i,
- Un vecteur de probabilités d'état initial π de longueur K tel que π_i représente la probabilité que $x_1 = s_i$

Sorties : La séquence la plus probable d'états $X = \{x_1, \cdots, x_T\}$

begin
  pour chaque *état* $s_i \in S$ **faire**
    $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
    $T_2[i, 1] \leftarrow 0$
  fin
  pour $i \leftarrow 2, 3, \cdots, T$ **faire**
    pour chaque *état* $s_j \in S$ **faire**
      $T_1[j, i] \leftarrow \max_k(T_1[k, i-1] \cdot A_{kj} \cdot B_{jy_i})$
      $T_2[j, i] \leftarrow \arg\max_k(T_1[k, i-1] \cdot A_{kj} \cdot B_{jy_i})$
    fin
  fin
  $z_T \leftarrow \arg\max_k(T_1[k, T])$
  $x_T \leftarrow s_{z_T}$
  pour $i \leftarrow T, T-1, \cdots, 2$ **faire**
    $z_{i-1} \leftarrow T_2[z_i, i]$
    $x_{i-1} \leftarrow s_{z_{i-1}}$
  fin
  retourner x
end

Algorithme 1: Viterbi (1967)

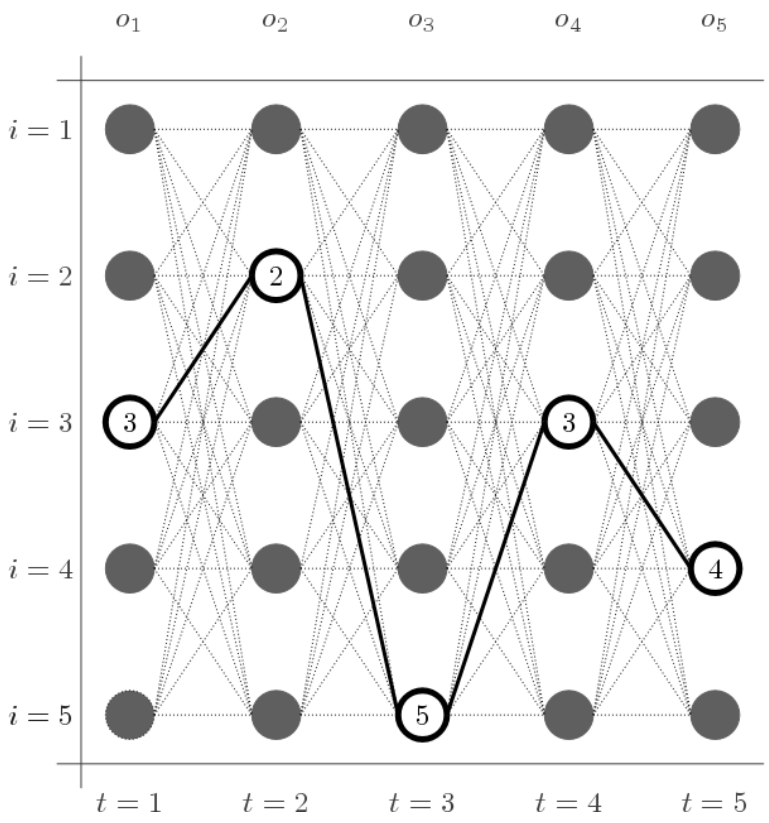

FIGURE 3.1 – Illustration de l'algorithme de Viterbi (licence Wikimedia Commons)

3.3 Modèle de Markov à entropie maximale

3.3.1 Principe d'entropie maximale

Dans certains cas, et particulièrement dans la tâche de détection d'entités nommés, il est également utile d'associer un contexte aux observations. L'ordre des mots (des observations), leur forme (contient une ou des majuscules) et les mots qui précèdent ou qui suivent, sont autant d'exemples d'informations qui ont leur importance. Prenons, en guise d'illustration, le mot « *ferme* ». Ce mot peut avoir 3 natures [8] :

1. Adjectif : Qui présente une certaine résistance à la pression, qui est consistant sans être dur.
2. Nom : Exploitation agricole qui a fait l'objet d'un bail à ferme ; p. ext. toute exploitation agricole.
3. Verbe : Manœuvrer (le ou les éléments d'une ouverture), de façon à priver de communication deux espaces. Conjugué à la 1re ou 3e personne du singulier au présent de l'indicatif.

Faisant abstraction de toutes les exceptions [9], admettons que la probabilité que « *ferme* » appartienne à l'une de ces catégories soit de 1.

$$p(nom) + p(adj) + p(verbe) = 1 \tag{3.11}$$

Sans aucune information contextuelle, le seul choix raisonnable pour déterminer la distribution des probabilités est celle qui est la plus uniforme, c'est-à-dire

8. http ://www.cnrtl.fr/definition/ferme
9. Une exception possible à cette règle est que quelqu'un soit nommé « Ferme ». Bien que peu probable, cela reste possible mais nous ne le considérons pas.

équivalente à toutes les parties (Berger *et al.*, 1996).

$$\forall i \in \{1, \cdots, n\} : p_i = \frac{1}{n} \tag{3.12}$$

Donc :

$$p(nom) = 1/3$$
$$p(adj) = 1/3$$
$$p(verbe) = 1/3$$

Cependant, en analysant un corpus de textes (annoté), admettons que nous découvrons que « ferme » est utilisé dans la moitié des cas en tant que verbe [10]. Nous avons alors le système d'équation suivant :

$$p(verbe) = 1/2$$
$$p(nom) + p(adj) = 1/2$$

Enfin, en appliquant une distribution uniforme pour les probabilités inconnues, nous obtenons la distribution suivante :

10. Aucune analyse allant dans ce sens n'a été effectuée, ce nombre, choisi au hasard, ne servira qu'à illustrer ce qui suit.

$$p(verbe) = 1/2$$
$$p(nom) = 1/4$$
$$p(adj) = 1/4$$

Ce principe, appelé le principe d'entropie maximale, revient à *respecter toutes les contraintes connues, mais autrement ne faire aucune supposition.*

La mise en contexte

En observant les mots autour de « *ferme* », on se rend compte qu'il existe une relation statistique entre la nature de ce mot et du ou des mots qui le précèdent. En effet, Si « *ferme* » est précédé d'un déterminant (comme « *une* »), alors « *ferme* » est, la plupart du temps, un nom. S'il est précédé d'un pronom (comme « je »), il sera probablement un verbe.

En analysant un corpus annoté, nous pouvons, pour chaque classe d'entité y, établir un contexte x, c'est-à-dire l'ensemble des mots qui entourent le mot à classifier comme y. Nous obtenons ainsi les paires $(x_1, y_1), \cdots, (x_N, y_N)$. Le but de cette analyse est de déterminer la distribution empirique $\tilde{p}$, définie par

$$\tilde{p}(x, y) = \frac{\text{nombre d'occurences de } (x, y) \text{ dans le corpus}}{N} \tag{3.13}$$

La fonction indicateur suivante détermine si le mot qui nous intéresse, « $ferme$ », de type « *nom* », est précédé ou pas d' « une » :

$$f(x,y) = \begin{cases} 1 \text{ si } \textit{une} \text{ précède } f\textit{erme} \text{ et que } y = \textit{nom} \\ 0 \text{ sinon} \end{cases} \tag{3.14}$$

Enfin, nous pouvons alors calculer la valeur attendue de f respectant la distribution empirique $\tilde{p}(x,y)$, comme suit :

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{3.15}$$

Nous obtenons ainsi la fréquence empirique à laquelle la fonction indicateur f est à 1 par rapport à toutes les observations. Une telle fonction est appelée un trait, et sa probabilité en ce qui concerne le modèle $p(y|x)$ peut être obtenue :

$$p(f) = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) \tag{3.16}$$

Enfin, nous pouvons contraindre notre modèle à respecter cette distribution, en posant $\tilde{p}(f) = p(f)$.

$$\sum_{x,y} \tilde{p}(x,y) f(x,y) = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) \tag{3.17}$$

Cependant, il existe beaucoup de modèles pouvant respecter les contraintes

posées, et le défi restant est donc de choisir le modèle qui respecte toutes les contraintes tout en restant le plus uniforme possible. Une mesure de l'uniformité d'une distribution conditionnelle $p(y|x)$ est donnée par Berger *et al.* (1996) :

$$H(p) \equiv -\sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x) \tag{3.18}$$

Il convient donc de choisir, parmi l'ensemble des modèles permis $p \in C$, celui qui a l'entropie maximale $H(p)$, soit :

$$p_* = \arg\max_{p \in C} H(p) \tag{3.19}$$

Nous n'entrons pas dans les détails du calcul de sélection du modèle le plus approprié, ceci n'étant pas nécessaire à la compréhension de la suite de ce mémoire. Cependant, les formules précédentes sont tirées de l'article de Berger *et al.* (1996). La démonstration complète peut être trouvée dans (Della Pietra *et al.*, 1997) et la méthode est celle des multiplicateurs de Lagrange. Nous retenons simplement la conclusion suivante :

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x,y)\right) \tag{3.20}$$

$$\Psi(\lambda) = -\sum_x \tilde{p}(x)\log Z_\lambda(x) + \sum_i \lambda_i \tilde{p}(f_i) \tag{3.21}$$

$Z_\lambda(x)$ représentant la constante de normalisation obtenue par la contrainte que

$\sum_y p_\lambda(y|x) = 1$ pour tout x :

$$Z_\lambda(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x, y)\right) \tag{3.22}$$

Enfin, par application des conditions de Kuhn-Tucker, il est admis qu'en trouvant une solution convenable au problème

$$\lambda^* = \arg\max_\lambda \Psi(\lambda), \tag{3.23}$$

Nous pouvons trouver une solution équivalente pour

$$p_* = \arg\max_{p \in C} H(p),$$

(Berger *et al.*, 1996)

3.3.2 Modèle de Markov à entropie maximale

Dans un modèle de Markov caché appliqué au traitement automatique du langage, une méthode pour étiqueter une séquence d'observations $o_1, \cdots, o_n$ consiste à associer des étiquettes $l_0, \cdots, l_n$ aux états $s_1, \cdots, s_n$ cachés du modèle. L'application de l'algorithme de Viterbi permet, pour une séquence d'observations, de déterminer la séquence d'états (et donc d'étiquettes) conformément au modèle.

McCallum *et al.* (2000a) proposent un nouveau modèle liant l'utilité du principe d'entropie maximale, et par extension de l'application de contraintes liées au contexte d'une observation, aux modèles de Markov. Ils affirment qu'il est intéres-

sant, dans le cas de l'étiquetage de séquences d'observations et notamment dans le domaine du traitement automatique du langage, de s'intéresser non seulement à l'état courant s, mais également à l'état précédent s'. Les probabilités de transitions et d'observations sont donc réunies au sein d'une même probabilité, $P(s|s', o)$, qui détermine la probabilité de l'état courant s compte tenu de l'état précédent s' et de l'observation courante o. En d'autre termes, c'est la probabilité de transition de l'état s' vers l'état s compte tenu de l'observation o. Les étiquettes qui sont liées à un état dans le cas d'un modèle de Markov caché ne sont plus directement liées aux états mais plutôt aux transitions entre les états.

Une fonction trait est choisie en fonction de son utilité perçue et va permettre de contraindre le modèle à respecter une distribution conforme au corpus d'entraînement.

$$f_{(b,s)}(o_t, s_t) = \begin{cases} 1 \text{ si } b(o_t) \text{ est vérifiée et } s = s_t \\ 0 \text{ sinon} \end{cases} \tag{3.24}$$

Le modèle doit en outre respecter la propriété d'entropie maximale suivante :

$$f_i(o_{t_k}, s_{t_k}) = \sum_{s \in S} P_{s'}(s|o_{t_k}) f_i(o_{t_k}, s) \tag{3.25}$$

Où $t_1, \cdots, t_{m_{s'}}$ est l'ordre des observations et où $s_{t_k} = s'$ est l'état qui est concerné par la probabilité de transition $P_{s'}$. La distribution respectant le principe

d'entropie maximale est donc :

$$P_{s'}(s|o) = \frac{1}{Z(o,s')} \exp\left(\sum_i \lambda_i f_i(o,s)\right) \tag{3.26}$$

Avec λ_i les paramètres du modèle, $Z(o,s')$ la constante de normalisation, de sorte que la somme de la distribution pour tous les états suivants s égale 1. (McCallum *et al.*, 2000a)

Biais d'étiquette

Les modèles Markoviens vus jusqu'à présent ont cependant une limitation qui peut se révéler problématique dans certains cas. En effet, le modèle est fortement biaisé lorsqu'il atteint des états qui possèdent peu de transitions. Considérons le cas extrême où un état ne possède qu'une seule transition : étant donné que la somme des probabilités pour atteindre le prochain état doit être égale à 1, la probabilité de la transition unique doit donc être à 1. Dès lors, une fois cet état atteint, le modèle ignore l'observation courante et la transition sera prise. Lafferty *et al.* (2001) nomment ce problème le biais d'étiquette. De manière générale, le décodage par Viterbi de la séquence entraîne le modèle à sélectionner les transitions dont les probabilités sont les plus élevées. Les états ayant peu de transitions ignorent donc l'observation.

3.4 Champs aléatoires conditionnels

Pour résoudre le problème du biais d'étiquette, Lafferty *et al.* (2001) proposent un nouveau modèle, les CRF (Conditionnal Random Fields), qui n'y sont pas sujets. Dans les CRF, les séquences d'observations et les séquences d'étiquettes

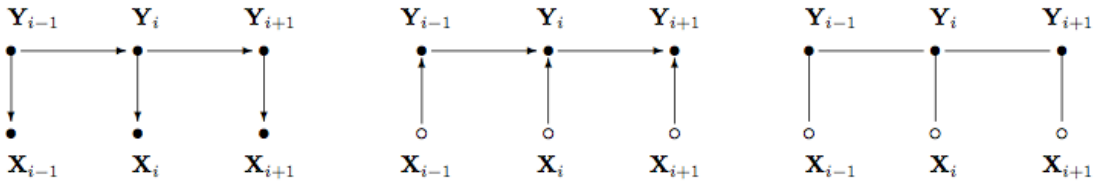

FIGURE 3.2 – Comparaison des modèles (HMM, MEMM, CRF). Les cercles vides sont des variables non générées par le modèle.
(Lafferty *et al.*, 2001)

correspondantes sont parfaitement associées. On calcule donc la probabilité globale de la séquence d'étiquettes pour une séquence d'observations donnée (Lafferty *et al.*, 2001).

Formellement, on définit X comme une séquence d'observations parmi l'ensemble des séquences d'observation possible, et Y une séquence d'étiquettes parmi l'ensemble des séquences d'étiquettes possible. La probabilité conditionnelle est donc $p(Y|X)$.

La figure 3.2 illustre la différence entre les modèles de Markov cachés, les modèles à entropie maximale et enfin les champs aléatoires conditionnels.

Bien que n'étant plus strictement un modèle de Markov, les CRF sont très proches de ces modèles car ils en sont très largement inspirés. On peut dont les qualifier de semi-Markoviens.

3.5 Limites des modèles Markoviens

3.5.1 Importance de la couverture lexicale

Les modèles Markoviens sont très intéressants : il « suffit » d'un corpus annoté pour construire un modèle fonctionnel. Cependant, bien qu'il est concevable qu'un tel corpus soit représentatif en termes de structures syntaxiques, son principal désavantage réside dans l'exhaustivité du lexique sur lequel il est entraîné. En effet, bien qu'un corpus peut être complet pour les mots communs d'un langage (noms communs, déterminants, prépositions, etc.), ou pour un certain domaine donné (par exemple pour la médecine), ce n'est certainement pas le cas pour les noms propres qui constituent l'essentiel des entités nommées.

Lors de l'entraînement d'un modèle Markovien, il est donc important de s'intéresser à la couverture lexicale fournie par le corpus d'entraînement. Si le corpus d'évaluation présente beaucoup d'observations et/ou de séquences d'observations inconnue du modèle, les performances ne seront pas optimales [11].

Les ressources linguistiques, notamment les dictionnaires à large couverture et les grammaires syntactico-sémantiques visent précisément à assurer la reconnaissance lexicale des unités non observées.

Par conséquent, la combinaison de ces deux approches semblent pertinente. C'est d'ailleurs cette hypothèse que nous défendons dans ce mémoire en intégrant les ressources linguistiques au sein du processus d'apprentissage d'un modèle statistique.

11. On peut observer ce phénomène simplement : si le corpus d'entraînement est également utilisé comme corpus d'évaluation, les performances mesurées seront excellentes (le modèle « apprend par cœur » le corpus d'entraînement), tandis que les performances sur des textes contenant des observations nouvelles ne seront pas au rendez-vous.

Parmi les travaux ayant utilisé une approche hybride, nous pouvons notamment citer (Constant *et al.*, 2011) qui intègrent des ressources linguistiques au sein d'un modèle CRF afin de créer un étiqueteur morphosyntaxique prenant en compte les unités polylexicales (Watrin, 2007). Ils atteignent ainsi de meilleurs résultats.

3.5.2 Modélisation des informations non locaux

Les modèles Markoviens présentés ici se basent sur des séquences d'observation où chaque observation repose sur des informations locaux (des traits du mot courant avec ou sans combinaison des traits des mots adjacents). Ce problème peut amener un même mot à être étiqueté de deux manières différentes au sein de la même séquence. Par exemple, dans la figure 3.3, il est intéressant d'encourager le système à étiqueter les deux apparitions de « Einstein » de la même manière, même si les informations locaux diffèrent.

Finkel *et al.* (2005) résolvent ce problème en utilisant un échantillonnage de Gibbs au lieu de l'algorithme Viterbi, en utilisant une fonction de pénalité lorsque il y a une incohérence d'étiquettes pour une même observation. Cependant, cette technique d'inférence peut augmenter le temps de traitement par un facteur de 30.

Pour résoudre ce problème de manière efficace, Krishnan et Manning (2006) utilisent un étiquetage en deux passes : la première permet d'extraire la plus grande partie de la structure tandis que la seconde permet de corriger les incohérences résiduelles.

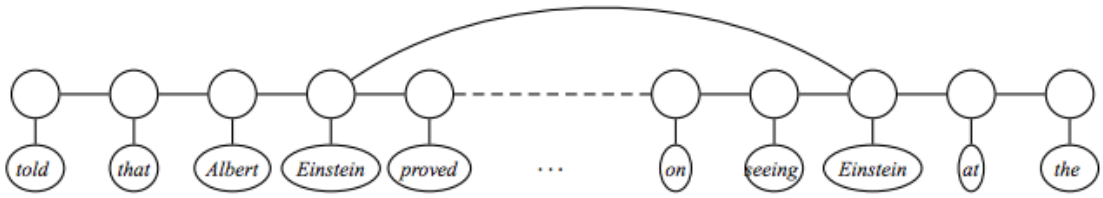

FIGURE 3.3 – Illustration du problème d'incohérence d'étiquetage : deux entités apparaissant avec des informations locales différentes, menant à un étiquetage incohérent. (Krishnan et Manning, 2006)

3.6 Autres techniques

L'étiquetage de séquences est un domaine de recherche relativement vieux et une grande quantité de techniques sont utilisées. L'étiquetage de séquences étant considéré comme un problème de classification, bon nombre de techniques de classification sont utilisées. Parmi celles-ci, les machines à vecteurs de support (Cortes et Vapnik, 1995), et en particulier les machines à vecteurs de support structuré (Tsochantaridis *et al.*, 2004) donnent également de très bons résultats. Les réseaux de neurones artificiels ont été utilisés pour faire de l'étiquetage morphosyntaxique (Schmid, 1994 ; Federici et Pirelli, 1994). Schmid (1994) montre également comment on peut utiliser des arbres de décision afin de construire un étiqueteur à partir d'un corpus de taille limitée.

Chapitre 4

Clustering

Le clustering est une technique appartenant aux méthodes d'apprentissage non supervisées. Elle vise à partitionner un certain nombre d'éléments dans un nombre restreint de *clusters* sur la base de certaines observations automatiques sur les éléments, également appelés traits. Dans ce chapitre, nous décrivons une technique de clustering très répandue, le k-means clustering, ainsi qu'une méthode permettant d'accélérer le processus de partitionnement. Enfin, nous parcourons les autres méthodes existantes.

4.1 k-means clustering

L'algorithme du k-means vise à partitionner n observations dans k partitions, avec $k \leq n$. Une observation est caractérisée par un vecteur de N réels, ses traits, et le partitionnement se base sur la méthode des moindres carrés.

Il faut donc trouver l'ensemble de partition $S = \{S_1, \cdots, S_k\}$ où, dans chaque

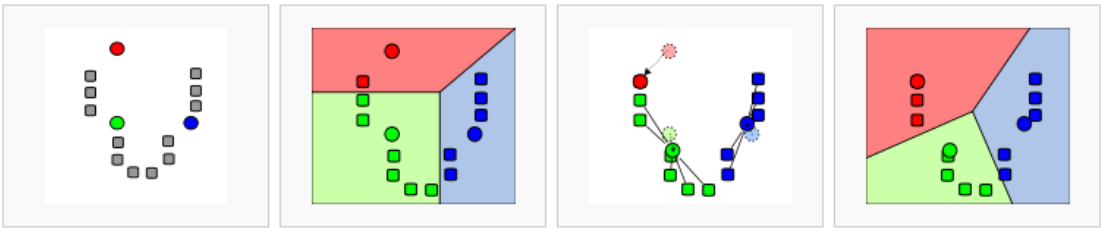

FIGURE 4.1 – Illustration de l'algorithme de k-means (Wikipedia Commons)

partition S_i, la somme des carrés de la distance euclidienne entre chaque observation de S_i, x_j et de la moyenne des observations μ_i est minimale :

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2 \tag{4.1}$$

Bien que ce problème soit NP-complet, il existe des heuristiques capables de converger vers un maximum local assez rapidement.

4.1.1 Algorithme

La figure 4.1 montre un exemple d'algorithme construisant des partitions :

1. Initialement, k « moyennes » sont aléatoirement posées dans l'espace à partitionner.
2. k partitions sont créées en fonction de la proximité de chaque « moyenne » par rapport aux observations.
3. La moyenne des observations au sein d'une partition devient la nouvelle « moyenne ».

On répète ensuite les étapes 2 et 3 jusqu'à ce qu'on arrête de converger.

4.1.2 Canopy clustering

Lorsque les données à partitionner sont trop nombreuses, que le nombre de traits est trop grand ou que beaucoup trop de partitions sont à trouver, l'algorithme peut prendre beaucoup de temps afin de converger vers des clusters desquels on peut tirer une information caractérisante. Pour alléger l'algorithme, McCallum *et al.* (2000b) ont conçu un pré-traitement connu aujourd'hui en tant que « Canopy Clustering ». Le fonctionnement, en deux temps, est relativement simple :

1. Il s'agit de faire un partitionnement « rapide » de toutes les informations. Ce partitionnement, bien que peu précis, permet d'établir des *canopy* qui rassemblent des informations fortement similaires.
2. Effectuer un partitionnement plus complexe au sein de chaque canopy.

McCallum *et al.* (2000b) montrent que cette approche permet même d'avoir de meilleurs résultats, de par sa nature à converger beaucoup plus rapidement.

Indexation inverse

Admettons que nous possédons un ensemble de textes, et que nous souhaitons partitionner ces textes en fonction de leur sujet. Chaque partition représente un sujet particulier et est caractérisé par des mots apparaissant fréquemment ensemble. Par exemple, les mots « Euros, Dollars et Investissement » peuvent faire partie d'une partition liée à la finance, tandis que les mots « Chat, Chien et Dauphin » feront partie d'une partition liée aux animaux.

Soient les mots $W = \{w_1, \cdots, w_N\}$ et les documents $D = \{d_1, \cdots, d_M\}$. Posons

ensuite :

$$\text{TF}(i,j) = \text{La fréquence du mot } w_i \text{ dans le document } d_j \tag{4.2}$$

$$\text{IDF}(i) = \log \frac{M}{\text{Le nombre de documents qui contiennent } i} \tag{4.3}$$

La matrice I mots-documents de taille $N \times M$ peut ainsi être créée telle que :

$$I(i,j) = \text{TF}(i,j) \times \text{IDF}(i) \tag{4.4}$$

Parallèlement, nous construisons une table de hachage dont les clés sont les mots w_i et les valeurs la liste des documents qui contiennent ce même mot w_i.

$$H = w_i \rightarrow D \backslash \{d_j \in D | TF(i,j) = 0\} \tag{4.5}$$

Les mots qui apparaissent dans quasi tous les documents peuvent être filtrés en posant la contrainte $IDF(i) < t$, où t représente le seuil au dessous duquel le mot sera admis dans la table de hachage. Pour filtrer les mots qui apparaissent dans 99% des documents, par exemple, ce seuil sera l'inverse de 99%, soit 100/99.

Enfin, pour chaque document d, nous pouvons chercher des documents similaires :

1. Pour chaque mot q_i du document d, récupérer, via H, la liste des documents R contenant le mot q_i.
2. Pour chaque document ainsi récupéré, calculer son score de similarité :

$$\text{sim}(d, r_j) = \frac{\sum_i I(i,j)}{|q \in d|} \tag{4.6}$$

3. Filtrer les documents trop peu similaires, en posant un seuil de similarité

minimal u.

La complexité en temps de l'opération 1 et 2 est, en moyenne, O(n) en nombre de mots du document choisi, ce qui est relativement rapide. Après l'étape 3, nous avons formé un canopy. L'opération peut être répétée autant de fois que nécessaire afin d'obtenir le nombre de partitions primaires souhaité. En dernier lieu, nous pouvons appliquer l'algorithme du k-means afin de trouver des partitions au sein même de ces partitions primaires.

4.2 Autres algorithmes

Il existe beaucoup de méthodes de clustering, par exemple le clustering hiérarchique qui consiste à créer des partitions progressivement en rassemblant les observations les plus proches d'abord, puis en rassemblant les différentes partitions en fonction de leur distance entre eux (Candillier, 2006). Il y a aussi les modèles basés sur la densité (Ester *et al.*, 1996), qui consiste à créer des partitions en fonction de la densité d'un certain espace d'observations. Ou encore en représentant les observations comme les nœuds d'un graphe et où les cliques représentent les partitions possibles ou que la somme des distances des arcs se trouve sous un seuil donné.

Des modèles plus spécialisés, comme LDA (Blei *et al.*, 2001), permettent également de partitionner des documents en fonction de sujets (non connus) en obtenant d'excellents résultats. Plus récemment, LSA (Dumais, 2004) permet de créer des partitions de mots en analysant un ensemble de documents, en partant du principe que des mots ayant un sens proche apparaîssent régulièrement dans des morceaux de textes similaires.

Chapitre 5

Evaluation des systèmes

L'évaluation sert à déterminer à quel degré un certain système détecte et classifie correctement les unités d'information contenues dans un texte. Au fil des campagnes d'évaluation organisées (MUC-6, IREX, CONLL, ACE), plusieurs méthodes d'évaluation ont été utilisés. Dans ce chapitre, nous parcourons ces méthodes et les comparons les unes aux autres.

5.1 Méthodes d'évaluation

Afin de pouvoir évaluer un système, nous devons disposer d'un corpus étiqueté qui n'est pas connu par le système à un quelconque point. Le corpus, sous sa forme non étiquetée, est traité par le système à évaluer. On compare ensuite le résultat du traitement au corpus original afin d'établir la correction sur deux dimensions : la précision et le rappel. Enfin, les systèmes peuvent être comparés sur la base d'une F-mesure, qui est la moyenne harmonique de la précision et du rappel du système.

TABLE 5.1 – Types de classes de sorties de l'évaluation MUC-6

Type	Résultat	Description
cor	Correct	La référence et la réponse sont les mêmes.
inc	Incorrect	La référence et la réponse diffèrent.
mis	Manquant	Il y a une référence mais pas de réponse.
spu	Surplus	Il y a une réponse sans référence.

TABLE 5.2 – Exemple d'évaluation d'un système pour MUC-6

Tag	Type	Text	Key Type	Response Type	Key Text	Response Text
ENAMEX	cor	cor	PERSON	PERSON	"Consuela Washington"	"Consuela Washington"
ENAMEX	cor	inc	PERSON	PERSON	"John Dingell"	"Washington"
ENAMEX	cor	inc	PERSON	PERSON	"Carter"	"Tim Wirt"
TIMEX	cor	cor	DATE	DATE	"01/19/93"	"01/19/93"
ENAMEX	mis	mis	PERSON		"Washington"	""
ENAMEX	spu	spu		ORGANIZATION	""	"Exchange"
ENAMEX	spu	spu		ORGANIZATION	""	"Old Executive Office"

5.1.1 MUC-6

La méthode d'évaluation du MUC-6 consiste d'abord en la classification de 4 types de résultats possibles. Dans le tableau 5.1, « référence » indique toute entité apparaissant dans le corpus d'évaluation, et « réponse » toute entité ayant été détectée par le système à évaluer. Pour chaque « détection », deux champs sont cotés : le *type* (organisation, personne et lieux, etc.) et le *texte*. Il est également à noter que l'ordre a de l'importance et que donc les éléments de référence et de réponse doivent être alignés afin d'obtenir du crédit. (Chinchor et Voorhees, 2001)

Par exemple, un système pourrait être évalué comme dans la table 5.2 :

Enfin, les formules suivantes sont utilisées afin d'obtenir un score :

$$N_{\text{pos}} = N_{\text{cor}} + N_{\text{inc}} + N_{\text{mis}} \tag{5.1}$$

$$N_{\text{act}} = N_{\text{cor}} + N_{\text{inc}} + N_{\text{spu}} \tag{5.2}$$

$$recall = \frac{N_{\text{cor}}}{N_{\text{pos}}} \tag{5.3}$$

$$precision = \frac{N_{\text{cor}}}{N_{\text{act}}} \tag{5.4}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{5.5}$$

Ce qui, dans l'exemple du tableau 5.2, donne un rappel de $\frac{6}{6+2+2} = 0.6$, une précision de $\frac{6}{6+2+4} = 0.5$. Enfin, la moyenne harmonique $F_1 = 0.55$.

5.1.2 IREX et CONLL

IREX et CONLL utilisent la même méthode d'évaluation, plus simple et plus restrictive que la méthode utilisée pour le MUC-6. La précision représente ici le pourcentage d'entités qui sont détectées entièrement correctement (type et texte), et le rappel est le pourcentage d'entités détectées par le système qui font partie de la solution de référence (Nadeau et Sekine, 2007).

$$N_{\text{detected}} = N_{\text{cor}} + N_{\text{inc}} + N_{\text{spu}} \tag{5.6}$$

$$N_{\text{solution}} = N_{\text{cor}} + N_{\text{inc}} + N_{\text{mis}} \tag{5.7}$$

$$precision = \frac{N_{\text{cor}}}{N_{\text{detected}}} \tag{5.8}$$

$$recall = \frac{N_{\text{cor}}}{N_{\text{solution}}} \tag{5.9}$$

L'exemple du tableau 5.2 donne donc comme précision $\frac{2}{6} = 0.33$ et comme rappel $\frac{2}{5} = 0.4$. Soit la moyenne harmonique $F_1 = 0.36$

5.1.3 ACE

Le système d'évaluation utilisé pour ACE apporte une notion de poids aux types d'entités. Chaque type est associé à une fonction de pondération. Certains types d'entités sont considérés plus importants que d'autres, et donc la performance d'un système est mesurée en fonction de son aptitude à détecter et classifier les types d'entités les plus importants. En outre, les mauvaises détections sont pénalisantes pour le système car la fonction de pondération apporte un facteur négatif (Doddington *et al.*, 2004).

$$\text{EDT}_{\text{value}} = \sum_i \text{value\_of\_entity}_i \tag{5.10}$$

$$\text{Value}_{\text{entity}} = \text{value\_of\_entity(entity)} \cdot \sum_m \text{mention\_value(mention}_m) \tag{5.11}$$

Cette approche pragmatique pour l'évaluation est intéressante au cas où nous souhaitons choisir entre les différentes solutions et où une importance est accordée à certain types d'entités plutôt qu'à d'autres. Cependant, la comparaison entre différents systèmes n'est possible que si les fonctions de pondération sont équivalentes (Nadeau et Sekine, 2007).

5.2 Validation

Pour s'assurer que les performances du modèle sont stables, on effectue une validation. On découpe ainsi le corpus en deux ensembles : le corpus d'entraînement et le corpus d'évaluation. Le corpus d'entraînement sert, comme son nom l'indique, à entraîner le modèle. On évalue ensuite le modèle entrainé en l'évaluant sur le corpus d'évaluation. Plusieurs approches existent pour choisir le corpus d'entraînement et le corpus d'évaluation. Elles sont décrites ci-dessous.

5.2.1 Validation naïve

L'approche la plus simple : on scinde simplement le corpus global en 2 parties, généralement de tailles inégales, le corpus d'entraînement étant généralement plus grand. Par exemple, si le corpus d'entraînement représente 60% du corpus global, alors le corpus d'évaluation est les 40% restants. L'avantage de cette approche est qu'elle ne requiert pas d'implémentation. Par contre, on risque un surentraînement du modèle : en ne validant le modèle que sur un corpus unique, on risque de biaiser celui-ci. Cela peut être un réel problème quand le corpus d'évaluation est trop petit et donc non représentatif de l'étendue des attentes que l'on a du modèle.

5.2.2 Validation croisée

Une approche un peu plus aboutie consiste à scinder le corpus en k parties égales. On peut ensuite utiliser $k-1$ parties comme corpus d'entraînement et la partie restante comme corpus d'évaluation. On répète ensuite ce processus jusqu'à ce que toutes les parties aient été exactement 1 fois le corpus d'évaluation. Enfin, l'évaluation globale est simplement la moyenne des k évaluations. Cette approche

requiert une implémentation, mais celle-ci est tellement simple qu'elle est utilisée presque exclusivement.

Conclusion

Dans cette première partie, nous avons défini la problématique et le cadre théorique du mémoire. Les tâches d'extraction d'informations sur lesquelles nous nous concentrons sont donc l'étiquetage morphosyntaxique et l'extraction d'entités nommées.

Nous avons présenté les différentes approches existantes, soient l'approche linguistique (symbolique) et les approches statistiques (apprentissage automatique).

Parmi les approches statistiques, nous distinguons trois méthodes différentes : supervisée, non supervisée ou semi-supervisée. La grande différence entre ces méthodes se situe tant dans les ressources nécessaires à leur application (corpus annoté entièrement, partiellement ou pas du tout) que dans leur approche du problème (classification parmi un ensemble de classes possibles ou plutôt classification conceptuelle).

Pour l'extraction d'informations, la méthode dominante aujourd'hui est la méthode supervisée. Dans ce cadre, nous avons décrit les modèles Markoviens de manière incrémentale. Nous avons vu que les modèles de Markov cachés peuvent être améliorés en y introduisant la notion de traits, permettant de contraindre un

modèle tout en maximisant son entropie. Enfin, les champs aléatoires conditionnels (CRF) améliorent les modèles de Markov à entropie maximale en résolvant un des problèmes auxquels ceux-ci sont sujets : le biais d'étiquette.

Les approches de Markov souffrent tout de même du problème de couverture lexicale. Nous pensons que c'est en combinant ces deux approches en une méthode hybride que les meilleurs résultats peuvent être obtenus, notamment en augmentant la couverture lexicale.

Parmi les méthodes non supervisées, nous avons montré le clustering k-means et une heuristique permettant d'augmenter la rapidité du processus : le canopy clustering.

Enfin, nous avons montré les méthodes d'évaluation et de validation présentées lors des différentes campagnes d'évaluation organisées pour cette tâche.

C'est en nous basant sur ces prérequis théoriques que nous avons pu développer le système d'extraction d'information automatique qui est décrit dans la seconde partie de ce mémoire.

Deuxième partie

Implémentation d'un système hybride d'étiquetage morphosyntaxique et d'extraction d'entités nommées

Introduction

Nous présentons ici le système hybride d'extraction d'informations automatique que nous avons développé. Nous nous intéressons particulièrement à deux tâches : l'étiquetage morphosyntaxique et l'extraction d'entités nommées. Dans ce cadre, nous parcourons et décrivons les ressources que nous avons obtenues ou développées, ainsi que leur structure interne et leur volumétrie. Ces ressources sont composées d'une part de corpus annotés, et d'autre part de ressources linguistiques (dictionnaires à large couverture et règles syntactico-sémantique).

Nous décrivons ensuite brièvement les outils utilisés, puis les étapes pour la création d'un système d'extraction automatique, qui consistent en :

1. L'ingénierie des traits.
2. La mise en place d'une stratégie d'entraînement.
3. L'entraînement du modèle en utilisant un algorithme choisi.
4. L'utilisation du modèle pour annoter des textes non connus.

L'évaluation et la validation du système sont obligatoires. Dans ce cadre, il convient d'utiliser des méthodes de validation comme le k-fold cross validation et une méthode d'évaluation afin de valider le modèle mais aussi de comparer les

performances d'un système par rapport à l'autre.

Chapitre 6

Présentation des ressources utilisées

Les ressources que nous utilisons pour la mise en place d'un système peuvent être divisées en deux catégories : les ressources statistiques et les ressources linguistiques. Les ressources statistiques sont construites manuellement afin d'entrainer un système de base (Baseline). Les ressources linguistiques sont une série de règles mises au point par des experts linguistes et prennent la forme de dictionnaires ou de grammaires.

6.1 Ressources statistiques

Par ressources statistiques, nous entendons les ressources construites manuellement afin d'entraîner un modèle statistique.

6.1.1 Morphosyntaxique

Nous utilisons le corpus du French Treebank [12] pour entraîner un étiqueteur morphosyntaxique. C'est un corpus annoté manuellement qui contient 3 types d'annotations :

1. Les annotations morphosyntaxique : Adjectif (A), adverbe (Adv), conjonction de coordination (CC), conjonction de subordination (CS), clitique (CI), déterminant (D), mot étranger (ET), interjection (I), nom commun (NC), nom propre (NP), préposition (P), préfixe (PREF), pronom (PRO), verbe (V) et ponctuation (PONCT).
2. Les annotations fonctionnelles : Sujet (SUJ), objet direct (OBJ), complément du sujet (ATS), complément d'objet direct (ATO), adjoint (MOD), complément indirect introduit par *à* (A-OBJ), complément indirect introduit par *de* (DE-OBJ), complément indirect introduit par une préposition (P-OBJ).
3. Les annotations en constituants : syntagme adjectival (AP), phrase adverbiale (AdP), phrase de coordination (COORD), phrase nominale (NP), préposition (PP), noyau verbal (VN), proposition infinitive (VPinf), proposition participale (VPpart), phrase (SENT), proposition finie (Sint, Srel, Ssub).

De plus, chaque mot est annoté de son lemme et éventuellement de ses sous-catégories. Le corpus est formé de fichiers XML qui contiennent un ensemble de phrases annotées. La figure 6.1 montre un exemple de phrase annotée.

Au niveau du volume, le corpus FTB (avec annotations fonctionnelles) représente environ 16.000 phrases pour plus de 470.000 mots, ce qui en fait le plus gros corpus francophone annoté syntaxiquement à ce jour.

12. http ://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php

```
▼<SENT nb="575">
 ▼<VPpart fct="MOD">
    <w cat="V" ee="V--Kms" ei="VKms" lemma="passer" mph="Kms" subcat="">Passé</w>
   ▼<NP>
      <w cat="D" ee="D-dem-ms" ei="Dms" lemma="ce" mph="ms" subcat="dem">cet</w>
      <w cat="N" ee="N-C-ms" ei="NCms" lemma="âge" mph="ms" subcat="C">âge</w>
    </NP>
  </VPpart>
  <w cat="PONCT" ee="PONCT-W" ei="PONCTW" lemma="," subcat="W">,</w>
 ▼<NP fct="SUJ">
    <w cat="D" ee="D-def-mp" ei="Dmp" lemma="le" mph="mp" subcat="def">les</w>
    <w cat="N" ee="N-C-mp" ei="NCmp" lemma="optimiste" mph="mp" subcat="C">optimistes</w>
  </NP>
 ▼<VN>
    <w cat="ADV" ee="ADV-neg" ei="ADV" lemma="ne" subcat="neg">ne</w>
    <w cat="V" ee="V--P3p" ei="VP3p" lemma="dépasser" mph="P3p" subcat="">dépassent</w>
  </VN>
  <w cat="ADV" ee="ADV-neg" ei="ADV" lemma="pas" subcat="neg">pas</w>
 ▼<NP fct="OBJ">
    <w cat="D" ee="D-def-mp" ei="Dmp" lemma="le" mph="mp" subcat="def">les</w>
    <w cat="A" ee="A-card-mp" ei="Amp" lemma="10" mph="mp" subcat="card">10</w>
    <w cat="N" ee="N-C-mp" ei="NCmp" lemma="%" mph="mp" subcat="C">%</w>
    <w cat="PONCT" ee="PONCT-W" ei="PONCTW" lemma=")" subcat="W">(</w>
   ▼<NP>
      <w cat="D" ee="D-def-ms" ei="Dms" lemma="le" mph="ms" subcat="def">le</w>
      <w cat="A" ee="A-qual-ms" ei="Ams" lemma="même" mph="ms" subcat="qual">même</w>
      <w cat="N" ee="N-C-ms" ei="NCms" lemma="nombre" mph="ms" subcat="C">nombre</w>
     ▼<Ssub>
        <w cat="C" ee="C-S" ei="CS" lemma="que" subcat="S">qu'</w>
       ▼<PP>
          <w cat="P" ee="P" ei="P" lemma="en">en</w>
         ▼<NP>
            <w cat="N" ee="N-card-fs" ei="NCfs" lemma="1980" mph="fs" subcat="card">1980</w>
          </NP>
        </PP>
      </Ssub>
    </NP>
    <w cat="PONCT" ee="PONCT-W" ei="PONCTW" lemma="(" subcat="W">)</w>
  </NP>
  <w cat="PONCT" ee="PONCT-S" ei="PONCTS" lemma="." subcat="S">.</w>
</SENT>
```

FIGURE 6.1 – Exemple d'annotation French Treebank

```
▼<DOCUMENT file="F050101A_0028">
   <ID>EXT010 3 GEN 0067 F BELGA-0055</ID>
   <CATEGORIES>ASIE/INDONESIE/CATASTROPHES/DIVERS</CATEGORIES>
   <SOURCE/>
   <DATE>2005-01-01</DATE>
  ▼<TITLE>
     Séisme en {Indonésie,.N+COUNTRY} (2): pas de danger de tsunami
   </TITLE>
  ▼<TEXT>
    ▼<S>
       {STRASBOURG,.N+TOWN} {01/01,.ADV+Time+PRPU} ( {AFP,.N+ORG} ) _ Tout danger de tsunami est écarté
       après le nouveau séisme qui s'est produit {samedi matin,.ADV+Time} au large de l' {île indonésienne
       de Sumatra,.N+HYDRO} , a indiqué à l' {AFP,.N+ORG} un {sismologue,.N+PROF} de l'{Observatoire des
       Sciences de la terre,.N+ORG} à {Strasbourg,.N+TOWN} . "Tout danger de raz-de-marée est écarté car il
       se serait déjà produit", a indiqué le {sismologue,.N+PROF} de permanence à l'observatoire./.
     </S>
     <S>LAR</S>
   </TEXT>
 </DOCUMENT>
```

FIGURE 6.2 – Exemple d'annotation d'entités nommées

6.1.2 Entités nommées

Pour l'entraînement de l'étiqueteur d'entités nommées, nous avons annoté manuellement un corpus de dépêches de presse. Les types d'entités annotés sont : les organisations (ORG), les personnes (PERS), les professions (PROF), les adresses (Address), les villes (TOWN), les régions (REGION), les pays (COUNTRY), les ensembles de pays (SUPRA), les étendues d'eau (HYDRO) et les entités temporelles (Time).

Pour ce qui est du volume, ce corpus est constitué de 625 dépêches pour un total d'environ 126.000 mots. Comme pour le corpus FTB, le corpus d'entités annoté est constitué de fichiers XML. Néanmoins ici, chaque fichier représente une dépêche et contient également certaines meta-informations sur la dépêche elle-même. La figure 6.2 montre un fichier du corpus. Les entités annotées respectent la notation Unitex (point 7.1.1, page 75) et se retrouvent donc entre accolades, suivies de leur type.

6.2 Ressources linguistiques

6.2.1 Morphosyntaxique

Plusieurs ressources linguistiques sont disponible :

Dictionnaire DELAF : Le dictionnaire DELAF [13] (Courtois *et al.*, 1990) est un dictionnaire fléchi de la langue française. Ce dictionnaire nous permet d'obtenir, pour une flexion d'un mot, ses catégories lexicales possibles ainsi que son lemme. Il contient plus de 680.000 entrées représentant plus de 100.000 lemmes pour les mots simples, et plus de 108.000 entrées pour plus de 80.000 lemmes pour les mots composés.

Splitter : C'est un ensemble de règles linguistiques sous forme de transducteurs [14] qui nous permettent de découper les textes en phrases.

Tokenisation : La tokenisation est un processus permettant de segmenter la phrase en mots. L'approche la plus simple tend à considérer qu'un mot est un ensemble de lettres encadré d'espaces ou de poncutations. Cependant, cette approche est trop naïve. Par exemple, le mot « aujourd'hui » ne devrait pas être considéré comme deux mots distincts, tandis que dans « l'habitude », il y a bien deux mots. Nous aurons donc recours à des règles précises, codées dans un transducteur, afin d'effectuer cette découpe correctement. Ces règles prennent en compte les propriétés de la langue française afin de séparer les mots correctement.

13. http ://infolingu.univ-mlv.fr/DonneesLinguistiques/Dictionnaires/telechargement.html
14. Un transducteur est une machine à états finis produisant une sortie.

6.3 Entités nommées

Les ressources linguistiques à disposition sont une série de règles lexico-syntaxiques mises en place manuellement. La figure 6.3 illustre une (des nombreuses) règle permettant de détecter une profession. Une telle règle nous permet de détecter des mots composés comme « sous-directeur », « directeur opérationnel », « directrice opérationnelle », etc. Quand un mot est indiqué entre chevrons, cela signifie que le graphe accepte toutes les flexions de ce lemme. Ces règles ont été développées avec l'outil Unitex, que nous présentons au point 7.1.1, page 75.

Pour développer ces règles, nous nous sommes basés sur un ensemble de grammaires qui fournissaient un haut taux de rappel, mais une précision imparfaite. Un élagage de ces grammaires a donc été effectué afin de réduire au maximum les erreurs de précision, quitte à diminuer le rappel. Nous pensons que la précision de ces règles est cruciale dans leur apport à l'approche hybride. Les grammaires ainsi élaguées ont un taux de précision de 97% pour un taux de rappel de 15% (évaluation faite sur le corpus d'entités nommées présenté plus haut).

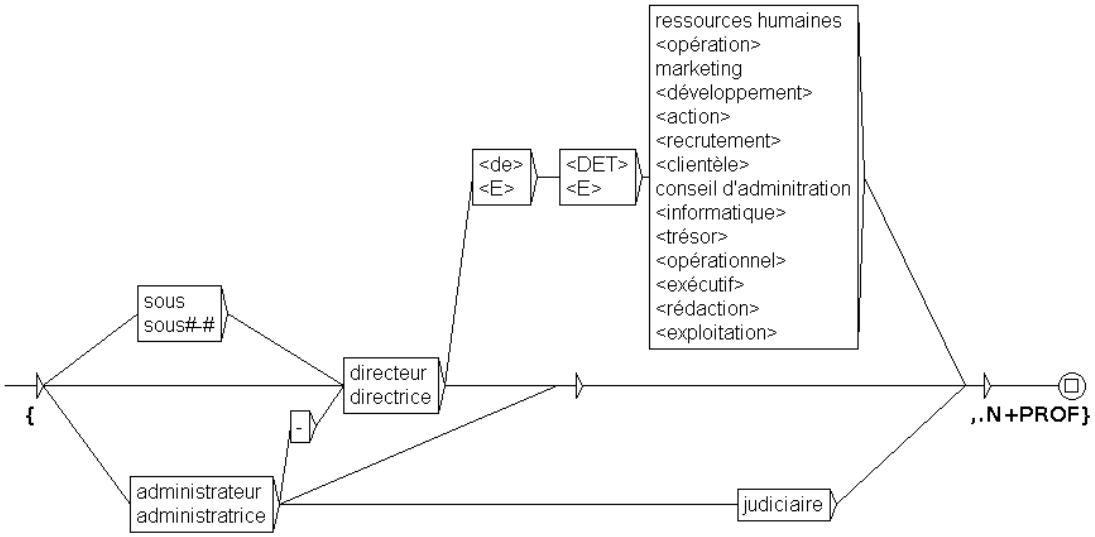

FIGURE 6.3 – Exemple de grammaire pour la détection d'une profession

Chapitre 7

Mise en place d'un étiqueteur morphosyntaxique et d'entités nommmées

Les ressources ayant été présentées, nous nous intéressons maintenant à la mise en place d'un système permettant l'étiquetage automatique tant en morphosyntaxe qu'en entités nommées. Nous commençons par présenter les outils utilisés puis décrivons les traits développés à partir des ressources linguistiques à notre disposition.

7.1 Présentation des outils

7.1.1 Unitex

Unitex [15] est une plateforme pour le traitement automatique du langage. Il nous permet d'appliquer des ressources linguistiques (dictionnaires et grammaires) sur

15. http ://www-igm.univ-mlv.fr/ unitex/

un texte afin d'en obtenir l'automate. Cet automate du texte représente un texte intégralement annoté (morphosyntaxiquement et sémantiquement) sur la base des ressources linguistiques fournies. L'exemple de l'ambigüité de « ferme » (montré au point 3.3.1, page 36) peut être résolu linguistiquement si nous disposons d'une règle qui correspond au prédicat : « si *ferme* est précédé d'un déterminant, alors *ferme* est un *nom commun* ». L'automate du texte contenant « ferme » n'aurait que « nom commun » pour le nœud « ferme ». Dans ce cas, la résolution de cette ambigüité repose sur la résolution de la catégorie lexicale du mot *précédent* ferme. Au cas où il persiste plusieurs possibilités, et qu'il y a donc ambigüité, Unitex les considère comme équiprobables.

7.1.2 CRFSuite

CRFSuite [16] est une implémentation des CRF, qui correspond à celles présentées par Lafferty *et al.* (2001); Sutton et McCallum (2010). Ce programme permet d'entraîner un modèle en choisissant une méthode parmi une multitude de choix possibles, en fournissant comme entrée des séquences d'observations associées à ledq :q rs traits. Le programme prend en entrée le corpus d'entraînement (comme i :wllustré sur la figure 7.1) et, facultativement, le corpus d'évaluation qui pourra être utilisé pour arrêter l'entraînement (par ot exemple, si le modèle ne converge plus vers une meilleure solution). De plus, un interfaçage `SWIG` est proposé afin de pouvoir entraîner et utiliser un modèle en utilisant un langage de programmation au choix. (Okazaki, 2007)

16. http ://www.chokkan.org/software/crfsuite/

```
B-NP    w[0]=An w[1]=A.P. pos[0]=DT pos[1]=NNP  __BOS__
I-NP    w[-1]=An  w[0]=A.P. w[1]=Green  pos[-1]=DT  pos[0]=NNP  pos[1]=NNP
I-NP    w[-1]=A.P.  w[0]=Green  w[1]=official pos[-1]=NNP pos[0]=NNP  pos[1]=NN
I-NP    w[-1]=Green w[0]=official w[1]=declined pos[-1]=NNP pos[0]=NN pos[1]=VBD
B-VP    w[-1]=official  w[0]=declined w[1]=to pos[-1]=NN  pos[0]=VBD  pos[1]=TO
I-VP    w[-1]=declined  w[0]=to w[1]=comment  pos[-1]=VBD pos[0]=TO pos[1]=VB
I-VP    w[-1]=to  w[0]=comment  w[1]=on pos[-1]=TO  pos[0]=VB pos[1]=IN
B-PP    w[-1]=comment w[0]=on w[1]=the  pos[-1]=VB  pos[0]=IN pos[1]=DT
B-NP    w[-1]=on  w[0]=the  w[1]=filing pos[-1]=IN  pos[0]=DT pos[1]=NN
I-NP    w[-1]=the w[0]=filing w[1]=.  pos[-1]=DT  pos[0]=NN pos[1]=.
O       w[-1]=filing  w[0]=.  pos[-1]=NN  pos[0]=.  __EOS__

B-NP    w[0]=The  w[1]=$  pos[0]=DT pos[1]=$  __BOS__
I-NP    w[-1]=The w[0]=$  w[1]=40-a-share pos[-1]=DT  pos[0]=$  pos[1]=JJ
I-NP    w[-1]=$ w[0]=40-a-share w[1]=proposal pos[-1]=$ pos[0]=JJ pos[1]=NN
I-NP    w[-1]=40-a-share  w[0]=proposal w[1]=values pos[-1]=JJ  pos[0]=NN pos[1]=VBZ
B-VP    w[-1]=proposal  w[0]=values w[1]=the  pos[-1]=NN  pos[0]=VBZ  pos[1]=DT
B-NP    w[-1]=values  w[0]=the  w[1]=company  pos[-1]=VBZ pos[0]=DT pos[1]=NN
I-NP    w[-1]=the w[0]=company  w[1]=at pos[-1]=DT  pos[0]=NN pos[1]=IN
B-PP    w[-1]=company w[0]=at w[1]=about  pos[-1]=NN  pos[0]=IN pos[1]=RB
B-NP    w[-1]=at  w[0]=about  w[1]=$  pos[-1]=IN  pos[0]=RB pos[1]=$
I-NP    w[-1]=about w[0]=$  w[1]=106.6  pos[-1]=RB  pos[0]=$  pos[1]=CD
I-NP    w[-1]=$ w[0]=106.6  w[1]=million  pos[-1]=$ pos[0]=CD pos[1]=CD
I-NP    w[-1]=106.6 w[0]=million  w[1]=.  pos[-1]=CD  pos[0]=CD pos[1]=.
O       w[-1]=million w[0]=.  pos[-1]=CD  pos[0]=.  __EOS__

B-NP    w[0]=A.P. w[1]=Green  pos[0]=NNP  pos[1]=NNP  __BOS__
I-NP    w[-1]=A.P.  w[0]=Green  w[1]=currently  pos[-1]=NNP pos[0]=NNP  pos[1]=RB
B-ADVP  w[-1]=Green w[0]=currently  w[1]=has  pos[-1]=NNP pos[0]=RB pos[1]=VBZ
B-VP    w[-1]=currently w[0]=has  w[1]=2,664,098  pos[-1]=RB  pos[0]=VBZ  pos[1]=CD
B-NP    w[-1]=has w[0]=2,664,098  w[1]=shares pos[-1]=VBZ pos[0]=CD pos[1]=NNS
I-NP    w[-1]=2,664,098 w[0]=shares w[1]=outstanding  pos[-1]=CD  pos[0]=NNS  pos[1]=JJ
B-ADJP  w[-1]=shares  w[0]=outstanding  w[1]=.  pos[-1]=NNS pos[0]=JJ pos[1]=.
O       w[-1]=outstanding w[0]=.  pos[-1]=JJ  pos[0]=.  __EOS__

B-NP    w[0]=Its  w[1]=stock  pos[0]=PRP$ pos[1]=NN __BOS__
I-NP    w[-1]=Its w[0]=stock  w[1]=closed pos[-1]=PRP$  pos[0]=NN pos[1]=VBD
```

FIGURE 7.1 – Exemple de corpus d'entraînement pour CRFsuite, repris du site de CRFsuite. Les étiquettes se trouvent à gauche et les traits de les traits de l'observation suivent. ___BOS___ et ___EOS___ représentent respectivement le début et la fin de la séquence.

7.2 Ingénierie des traits

Dans notre application, un trait est le résultat d'une fonction à valeur discrète dont l'entrée est un mot. Par exemple, un trait peut être le résultat de la fonction booléenne « contient-un-chiffre » ($true$ ou $false$), mais aussi la fonction « préfixe » dont la valeur est une chaîne de caractères [17]. Un trait peut également être appliqué non pas au mot courant, mais aux mots qui entourent ce dernier. Le décalage par rapport au mot courant est déterminé par un entier, appelé l'*offset*. L'*offset* 0 correspondant au mot courant, -1 au mot précédent, 1 au mot suivant, etc.

Nous avons vu dans le chapitre 3 qu'il est nécessaire de maximiser la couverture lexicale afin d'obtenir le meilleur modèle possible. C'est dans cette optique que nous avons choisi les traits présentés dans cette section. D'une part les traits morphologiques afin de maximiser la couverture lexicale induite par les ressources statistiques (corpus), et de l'autre les traits grammaticaux afin d'introduire un biais, dont l'exactitude est quasi certaine, dans le modèle.

Dans la suite de ce mémoire, nous utiliserons les termes « trait » et « fonction trait » de manière interchangeable, le contexte ne permet qu'une seule interprétation.

7.2.1 Types de traits

2 types de traits peuvent se distinguer : ceux dont l'ensemble des valeurs est borné et ceux dont la valeur est une transformation appliquée à l'observation.

17. Pour « chat », les préfixes possibles sont donc « c », « ch », « cha » ou « chat ».

Traits à valeurs bornées

Les traits dont l'ensemble des valeurs est bornée sont généralement des traits booléens et leur valeur peut être soit « true », soit « false ». Chacun de ces générateurs crée donc une paire de fonctions traits :

$$f_{(b,s)}(o_t, s_t) = \begin{cases} 1 \text{ si } b(o_t) \text{ est vérifiée et } s_t = NC \\ 0 \text{ sinon} \end{cases} \tag{7.1}$$

Par exemple, pour le trait « hasHyphen » [18] et l'étiquette « NC » [19], les fonctions générées sont comme suit :

$$f^1_{(hasHyphen,NC)}(o_t, s_t) = \begin{cases} 1 \text{ si } hasHyphen(o_t) = true \text{ et } s_t = NC \\ 0 \text{ sinon} \end{cases}$$

$$f^2_{(hasHyphen,NC)}(o_t, s_t) = \begin{cases} 1 \text{ si } hasHyphen(o_t) = false \text{ et } s_t = NC \\ 0 \text{ sinon} \end{cases}$$

Traits à valeurs illimitées

L'exemple le plus simple de trait à valeur illimitée est le trait « mot(courant) ». En effet, chaque mot du corpus mène le générateur de traits à créer une fonction trait. Le trait « mot » peut donc avoir, théoriquement, un nombre illimité de

18. La liste des traits ainsi que leur description est donnée dans la section suivante.
19. Nom commun

valeurs différentes. Le générateur crée autant de fonctions traits qu'il y a de valeurs distinctes et correspond à la taille du vocabulaire utilisé.

$$f^1_{(word,NC)}(o_t, s_t) = \begin{cases} 1 \text{ si } word(o_t) = y_1 \text{ et } s_t = NC \\ 0 \text{ sinon} \end{cases}$$

$$\vdots$$

$$f^n_{(word,NC)}(o_t, s_t) = \begin{cases} 1 \text{ si } word(o_t) = y_n \text{ et } s_t = NC \\ 0 \text{ sinon} \end{cases}$$

Où $\{y_1, \cdots, y_n\}$ représentent les valeurs possibles du trait.

7.2.2 Générateurs de traits

Traits morphologiques

Les traits morphologiques ne se basent que sur la forme du mot. Ceux-ci sont les suivants :

Traits à valeurs bornées

1. Contient des traits d'unions : *hasHyphen(offset)*
2. Contient un chiffre : *hasDigit(offset)*
3. Est tout en capitales : *allUppercase(offset)*
4. Commence par une majuscule : *isCapitalized(offset)*
5. Début de la phrase : *BOS(offset)*

Traits à valeurs illimitées

1. Le mot : *w[offset]*
2. Préfixe de longueur l : *prefix(l, offset)*
3. Suffixe de longueur l : *suffix(l, offset)*
4. Forme du mot [20] : $shape(offset)$

Traits grammaticaux

Classe d'ambiguïté morphosyntaxique *ac(offset)* : les mots de la langue française peuvent faire partie de plusieurs catégories lexicales en fonction du contexte dans lequel ils sont utilisés. Nous avons vu que le mot *ferme*, par exemple, pouvait être un nom commun, un verbe ou un adjectif. Le trait *ac* représente ces ambiguïtés possibles : c'est simplement la concaténation des différentes catégories lexicales auxquelles peut appartenir un mot. L'algorithme pour récupérer ces classes consiste en la consultation d'un dictionnaire.

Classe d'ambiguïté sémantique *sac(offset)* : lors de l'application des grammaires, certains mots peuvent être classifiés comme ambigus. Le trait *sac* reprend donc, comme pour le trait *ac*, cette ambiguïté, pour les classes sémantiques uniquement, c'est-à-dire les types que peuvent prendre une entité nommée.

Appartenance à au moins un type d'entité *containsFeature(offset)* : le trait *containsFeature* est un trait booléen qui renvoie « true » si l'application des grammaires classifie le mot comme une entité (peu importe si une ambiguïté sémantique existe).

20. Cette fonction remplace tous les caractères alphabétique par la lettre « x » en respectant la casse, toutes les ponctuations (ex : trait d'union) par « - » et tous les chiffres par « $ ».

7.2.3 Étiqueteur morphosyntaxique

L'étiqueteur morphosyntaxique est entraîné, à partir du corpus FTB, sur la base d'une combinaison de traits morphologiques et de traits d'ambigüité grammaticale. La liste ci-dessous reprend les traits que l'on utilise afin d'entraîner notre étiqueteur morphosyntaxique à partir du corpus FTB.

1. *w(0)*
2. *w(-1)+w[0]*
3. *w(0)+w[1]*
4. *w(-1)+w[1]*
5. *lowercase(0)*
6. *hasHyphen(0)*
7. *hasDigit(0)*
8. *allUppercase(0)*
9. *isCapitalized(0)*
10. *BOS(0)*
11. $\forall n = \{1, \cdots, 4\}$, *prefix(n,0)*
12. $\forall n = \{1, \cdots, 4\}$, *suffix(n,0)*
13. $\forall n = \{-2, \cdots, 2\}$, *ac(n)*

7.2.4 Étiqueteur d'entités nommées

L'étiqueteur d'entités nommées est entraîné sur la base des même traits que l'étiqueteur morphosyntaxique. Nous y ajoutons cependant quelques traits, dont l'étiquette morphosyntaxique (*POS(offset)*), ainsi que les traits sémantiques issus des ressources linguistiques (*sac(offset)*, *containsFeature(offset)*).

1. *w(-1)*
2. *w(1)*
3. *shape(-1)*
4. *shape(1)*
5. *POS(0)*
6. *POS(-1)*
7. *POS(1)*
8. *POS(-1) + POS(0)*
9. *POS(0) + POS(1)*
10. $\forall n = \{-2, \cdots, 2\}$, *sac(n)*
11. *containsFeature(0)*

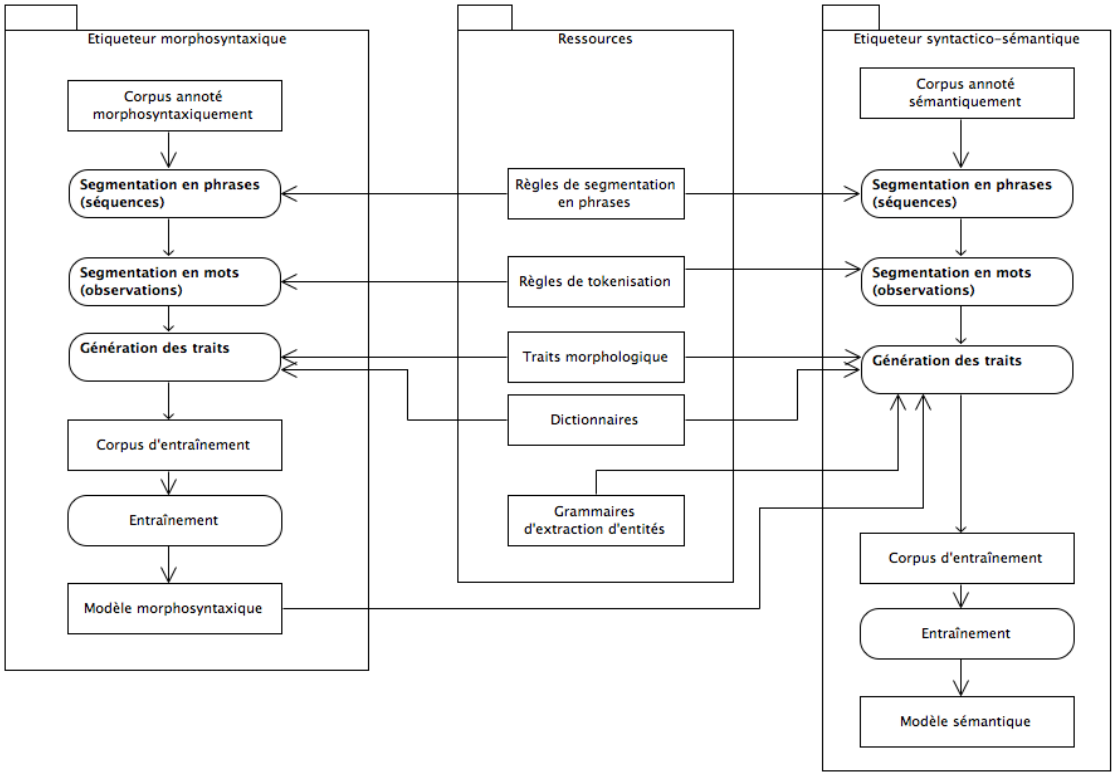

FIGURE 7.2 – Architecture de base : les processus en gras sont également effectués lors de l'application du modèle entrainé.

7.3 Architecture de base

L'architecture de notre étiqueteur est basée sur 3 éléments : l'étiquetage morphosyntaxique, les ressources et l'étiqueteur sémantique (entités nommées). La figure 7.2 illustre les grandes étapes pour l'entraînement d'un modèle. Les carrés représentent des ressources ou des résultats, et les ovales des processus. Chaque processus prend ainsi une ou plusieurs ressources en entrée et produit une ressource en sortie. Cette ressource peut à son tour être utilisée comme entrée dans un autre processus.

Chapitre 8

Évaluation

Afin d'évaluer et de valider les performances du système, nous avons procédé à une évaluation qui nous permettra de juger de l'apport de chacune des ressources au système indépendamment les unes des autres.

8.1 Méthode d'évaluation

Nous avons utilisé la méthode d'évaluation présentée lors de l'IREX/CONLL (point 5.1.2, page 57) afin d'évaluer les performances du système développé. Cette évaluation consiste à déterminer 4 mesures :

- N_{cor} : le nombre d'étiquettes correctes (détection et classification conforme au corpus d'évaluation).
- N_{inc} : le nombre d'étiquettes incorrectes (détection partielle d'un mot composé ou classification incorrecte par rapport au corpus).
- N_{spu} : le nombre d'étiquettes superflues (détection et classification sans référence dans le corpus).

	Précision	Rappel	F_1
Baseline	97,17%	93,06%	95,07%
AC	97,79%	94,44%	96,09%

TABLE 8.1 – Evaluation de l'étiqueteur morphosyntaxique

– N_{mis} : le nombre d'étiquettes manquantes (non détection alors qu'il y a une référence dans le corpus).

Nous mesurons ensuite la précision et le rappel de chaque système comme suit :

$$
\begin{aligned}
N_{\text{detected}} &= N_{\text{cor}} + N_{\text{inc}} + N_{\text{spu}} \\
N_{\text{solution}} &= N_{\text{cor}} + N_{\text{inc}} + N_{\text{mis}} \\
precision &= \frac{N_{\text{cor}}}{N_{\text{detected}}} \\
recall &= \frac{N_{\text{cor}}}{N_{\text{solution}}} \\
F_1 &= \frac{2 \cdot precision \cdot recall}{precision + recall}
\end{aligned}
$$

Chaque système a été évalué selon une validation croisée à dix échantillons ($k = 10$) (point 5.2.2, page 59).

8.2 Étiqueteur morphosyntaxique

Nous avons procédé à l'entraînement de 2 étiqueteurs morphosyntaxiques. Le premier ne se base que sur les traits morphologiques (Baseline) tandis que le second est enrichi par le trait *ac(offset)*, qui est calculé à partir des dictionnaires morphosyntaxiques de la langue française (DELAF), comme vu précédemment.

	Précision	Rappel	F_1
Baseline	78,93%	74,31%	76,55%
POS	78,56%	74,88%	76,68%
AC	79,28%	75,83%	77,52%
SAC	**81,67%**	76,52%	79,01%
POS+SAC	81,23%	77,49%	79,32%
AC+SAC	81,09%	77,72%	79,37%
POS+AC+SAC	81,07%	**78,01%**	**79,51%**

TABLE 8.2 – Evaluation de l'étiqueteur d'entités nommées. Le maximum de chaque colonne est en gras.

Nous pouvons observer une amélioration notable tant en terme de précision qu'en terme de rappel, comme le montre le tableau 8.1.

8.3 Étiqueteur d'entités nommées

Comme pour l'étiqueteur morphosyntaxique, nous avons procédé progressivement à l'ajout de ressources linguistiques afin de mettre en évidence l'apport de chacune de celles-ci. Le tableau 8.2 reprend les performances de chaque système ainsi construit. Le trait *ac* est utilisé, comme précédemment, pour désigner l'ambiguïté morphosyntaxique, tandis que le trait *sac* désigne l'ambiguïté sémantique et est apporté par les grammaires d'extraction d'entités nommées. Nous pouvons noter que l'apport des ressources linguistiques est toujours positif en terme de F-mesure.

Cependant, bien que l'apport individuel de chacune des ressources (*AC*, *SAC*) améliore tant la précision que le rappel, leur combinaison est légèrement moindre en terme de précision que celle amenée par l'utilisation des grammaires seulement. Par contre, cette combinaison augmente largement le rappel, ce qui mène à une F-mesure supérieure. Ceci nous indique clairement la pertinence de l'introduction

des ressources linguistiques afin d'augmenter la couverture lexicale, donnant lieu à un système plus performant.

Enfin, nous avons mesuré l'apport des étiquettes morphosyntaxique, en utilisant l'étiqueteur morphosyntaxique précédemment entraîné, et le résultat est semblable à l'apport des ressources linguistiques : une augmentation sensible du rappel et une petite réduction de la précision qui mènent à une augmentation, bien que relative, de la F-mesure.

Conclusion

L'approche symbolique et l'approche statistique sont toutes les deux pertinentes et surtout complémentaires.

Après avoir présenté les ressources, tant statistiques que linguistiques, obtenues ou développées pour chacune de ces deux approches, nous avons présenté une liste de traits pertinents afin de pouvoir tirer profit tant des ressources statistiques que linguistiques.

Pour atteindre ces objectifs, nous avons conçu et développé un programme qui prend en entrée les différentes ressources et produit un modèle hybride qui sera utilisable pour résoudre la tâche d'étiquetage morphosyntaxique et celle d'étiquetage d'entités nommées. Cette architecture nous permet également d'utiliser un modèle comme trait pour un autre système, comme c'est le cas par exemple avec l'apport de l'étiquette morphosyntaxique pour l'étiquetage d'entités nommées.

Enfin, nous avons montré que cette approche hybride pour l'extraction d'information permet d'obtenir de meilleures performances que l'une ou l'autre approche utilisée seule. En particulier, les grammaires permettent d'augmenter significativement la précision tandis que les dictionnaires à large couverture améliorent le

rappel. C'est la combinaison de toutes les ressources qui mène aux meilleurs résultats via une approche hybride.

Conclusion générale

Dans ce mémoire, nous avons défini la problématique de la tâche d'étiquetage morphosyntaxique et d'entités nommées. Nous avons étudié les différentes méthodes et approches existantes au problème et avons par ce biais développé des ressources et un système répondant à celui-ci. Nous avons évalué l'apport de chaque ressource linguistique au sein d'un modèle probabiliste et avons pu créer un système d'étiquetage compétitif.

Dans cette conclusion finale, nous amenons quelques applications possibles d'un tel système, en particulier au sein des moteurs de recherche. Enfin, nous concluons ce mémoire par certaines perspectives d'améliorations encore possibles.

Applications possibles dans les moteurs de recherche

L'étiquetage en entités nommées est intéressante pour les moteurs de recherche et il existe une multitude d'utilisations pertinentes de ces informations.

Par exemple, un moteur de recherche peut tirer profit d'un index des entités

nommées afin de proposer à l'internaute une auto-complétion intelligente : l'introduction du mot « Appl » dans la barre de recherche peut révéler une liste déroulante permettant la sélection de « Apple Inc. **(Société)** ».

Un index des entités permet également de retrouver rapidement les entités apparaissant régulièrement ensemble dans des documents. Cette application peut mettre en évidence certaines relations entre différentes entités. Pour les moteurs de recherche, cela peut se traduire par la suggestion automatique de nouveaux critères de recherche, ou même l'incorporation automatique de documents ne mentionnant pas directement l'entité recherchée mais contenant un grand nombre d'entités statistiquement pertinentes. Par exemple, une forte relation fréquentielle existe entre Google et Microsoft. Une recherche comme « concurrents de Microsoft » pourrait automatiquement inclure les documents concernant Google. Ceci permet aussi d'effectuer des recherches sémantiques comme « restaurants à Louvain-La-Neuve ».

Perspectives d'améliorations

Bien qu'il existe des systèmes et des approches performantes, les tâches d'étiquetage morphosyntaxique et d'entités nommées ne peuvent pas encore être considérées comme résolues. L'approche hybride que nous présentons permet d'obtenir de meilleurs résultats, mais ceux-ci restent perfectibles.

La manière la plus simple et qui ne requiert pas de développement informatique supplémentaire serait d'augmenter le volume des ressources : annoter un plus gros volume de textes ou développer plus de règles linguistiques (principalement des grammaires).

Nous pouvons également analyser les classifications une à une et déterminer une

typologie d'erreurs afin de convenir d'une méthode de correction automatique après le traitement par le système. Des travaux ont déjà été effectué dans ce domaine et les résultats sont prometteurs (Krishnan et Manning, 2006).

Une autre piste intéressante est d'effectuer un clustering sur base de synsets, comme le font Alfonseca et Manandhar (2002), sur un corpus plus volumineux afin de permettre l'ajout d'un trait représentant le cluster de chaque mot. Similairement, nous pouvons également utiliser des techniques comme LSA et LDA.

Bibliographie

Enrique Alfonseca et Suresh Manandhar : An unsupervised method for general named entity recognition and automated concept discovery. *In Proceedings of the 1st International Conference on General WordNet, Mysore, India*, pages 34–43, 2002.

Leonard E Baum, Ted Petrie, George Soules et Norman Weiss : A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Adam L Berger, Vincent J Della Pietra et Stephen A Della Pietra : A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

David M Blei, Andrew Y Ng et Michael I Jordan : Latent dirichlet allocation. *In Advances in neural information processing systems*, pages 601–608, 2001.

Eric Brill : *A corpus-based approach to language learning*. Thèse de doctorat, Citeseer, 1993.

Laurent Candillier : *Contextualisation, visualisation et évaluation en apprentissage non supervisé*. Thèse de doctorat, Université Charles de Gaulle-Lille III, 2006.

Nancy Chinchor, Patty Robinson et Erica Brown : Hub-4 named entity task definition version 4.8. *Available by ftp from www.nist.gov/speech/hub4_98*, 1998.

Nancy Chinchor et Ellen Voorhees : The message understanding conference scoring software user's manual, 01 2001. URL `http://www-nlpir.nist.gov/related_projects/muc/muc_sw/muc_sw_manual.html`.

Nancy A Chinchor : Overview of muc-7/met-2. 1998.

Matthieu Constant, Isabelle Tellier, Denys Duchier, Yoann Dupont, Anthony Sigogne, Sylvie Billot *et al.* : Intégrer des connaissances linguistiques dans un crf : application à l'apprentissage d'un segmenteur-étiqueteur du français. *TALN2011*, 1, 2011.

Corinna Cortes et Vladimir Vapnik : Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Blandine Courtois, Max Silberztein Ladl *et al.* : Dictionnaires électroniques du français. *Langue française*, 87(1):3–4, 1990.

Stephen Della Pietra, Vincent Della Pietra et John Lafferty : Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393, 1997.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel et Ralph Weischedel : The automatic content extraction (ace) program–tasks, data, and evaluation. *In Proceedings of LREC*, volume 4, pages 837–840. Citeseer, 2004.

Susan T Dumais : Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu : A density-based algorithm for discovering clusters in large spatial databases with noise. *In KDD*, volume 96, pages 226–231, 1996.

Jenny Rose Finkel, Trond Grenager et Christopher Manning : Incorporating non-local information into information extraction systems by gibbs sampling. *In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

Barbara B Greene et Gerald M Rubin : *Automatic grammatical tagging of English*. Department of Linguistics, Brown University, 1971.

Ralph Grishman et Beth Sundheim : Message understanding conference-6 : A brief history. *In Proceedings of COLING*, volume 96, pages 466–471, 1996.

Vijay Krishnan et Christopher D Manning : An effective two-stage model for exploiting non-local dependencies in named entity recognition. *In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1121–1128. Association for Computational Linguistics, 2006.

John Lafferty, Andrew McCallum et Fernando CN Pereira : Conditional random fields : Probabilistic models for segmenting and labeling sequence data. 2001.

Andrew McCallum, Dayne Freitag et Fernando CN Pereira : Maximum entropy markov models for information extraction and segmentation. *In ICML*, pages 591–598, 2000a.

Andrew McCallum, Kamal Nigam et Lyle H Ungar : Efficient clustering of high-dimensional data sets with application to reference matching. *In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000b.

David Nadeau et Satoshi Sekine : A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

Naoaki Okazaki : Crfsuite : a fast implementation of conditional random fields (crfs), 2007. URL `http://www.chokkan.org/software/crfsuite/`.

Lawrence R Rabiner : A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Lisa F Rau : Extracting company names from text. *In Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, volume 1, pages 29–32. IEEE, 1991.

Tjong Kim Sang et F Erik : Introduction to the conll-2002 shared task : Language-independent named entity recognition. *In proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics, 2002.

Diana Santos, Nuno Seco, Nuno Cardoso et Rui Vilela : Harem : An advanced ner evaluation contest for portuguese. *In Proceedings of LREC*, pages 1986–1991, 2006.

Helmut Schmid : Probabilistic part-of-speech tagging using decision trees. *In Proceedings of international conference on new methods in language processing*, volume 12, pages 44–49. Manchester, UK, 1994.

Satoshi Sekine et Hitoshi Isahara : Irex : Ir and ie evaluation project in japanese. *In Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 1475–1480, 2000.

Yusuke Shinyama et Satoshi Sekine : Named entity discovery using comparable news articles. *In Proceedings of the 20th international conference on Computational Linguistics*, page 848. Association for Computational Linguistics, 2004.

Charles Sutton et Andrew McCallum : An introduction to conditional random fields. *arXiv preprint arXiv :1011.4088*, 2010.

Erik F Tjong Kim Sang et Fien De Meulder : Introduction to the conll-2003 shared task : Language-independent named entity recognition. *In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims et Yasemin Altun : Support vector machine learning for interdependent and structured output spaces. *In Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

Andrew Viterbi : Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13 (2):260–269, 1967.

Patrick Watrin : Collocations et traitement automatique des langues. *In 26ème Colloque international sur le lexique et la grammaire (LGC'07)*, numéro 1, pages 191–198, 2007.